

# A comparison of structural optimization techniques with extension towards multi-objective problems

by

Theodorus Ernst Müller



*Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering in the Faculty of Engineering at Stellenbosch University*

Supervisor: Mr. E. van der Klashorst

Co-supervisor: Dr. G.C. van Rooyen

December 2017

## Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

---

T.E. Müller

December 2017

Copyright © 2017 Stellenbosch University  
All rights reserved

## Abstract

Structural optimization is becoming an integral part of the modern structural design process in the search to yield more economical structures. The optimization of structures is typically performed with the objective to minimize weight or displacement primarily for cost reasons. Normally only one objective is considered, but methods enabling the consideration of multiple objectives have been developed. With respect to truss and frame structures, there are three well-known aspects which can be considered during the optimization process, namely, member sizing, shape and topology. These aspects refer to the size of the structure's members, the internal member configuration and its nodal positioning respectively. During the optimization, these aspects can be considered individually, simultaneously or sequentially, although typically only member size is considered due to its simplicity. This study aims to quantify the weight reduction in the resulting truss structure by applying a more complex optimization approach such as considering the three aspects simultaneously. Furthermore, this study also aims to determine whether or not a meaningful weight reduction can be achieved by adjusting the prescribed deflection limit of a frame structure whose maximum deflection can be regarded as non-critical, for example a rural warehouse.

These aims are achieved by researching both general and structural optimization as well as available algorithms for successfully optimizing a structure. Software is developed to find solutions to both single- and multi-objective optimization problems. The software is used to optimize various truss problems found in literature by considering different combinations of the aforementioned structural aspects. The software is also used to optimize a selection of frame structures in a multi-objective

manner by minimizing both weight and displacement.

It is concluded that a 22% more economical solution can be found by considering the three aspects simultaneously as opposed to considering only member size. From the frame structures considered in this study, it is concluded that the majority of the structure's weight can be reduced before the deflection limit is reached. Therefore, an increase in the displacement limit is not required.

## Opsomming

Strukturele optimering vorm 'n belangrike deel van die moderne struktuurontwerp proses om ten einde meer ekonomiese strukture te verkry. Strukture word tipies geoptimeer met die doel om die betrokke struktuur se gewig of verplasing te minimeer. Alhoewel metodes ontwikkel is vir die oorweging van meer as een doel, word normaalweg slegs een oorweeg. Ten opsigte van vakwerk en raam strukture is daar drie aspekte wat oorweeg kan word tydens die optimeringsproses, naamlik elementgrootte, vorm en topologie. Hierdie aspekte verwys respektiewelik na die grootte van elemente, die interne element konfigurasie en die posisionering van knooppunte. Tydens die optimeringsproses kan hierdie aspekte individueel, gelyktydig of agtereenvolgend oorweeg word, alhoewel meestal slegs die element groottes oorweeg word weens die eenvoudigheid daarvan. 'n Doelwit van hierdie studie is om die verbetering in die vakwerk struktuur wanneer 'n meer komplekse optimeringstegniek toegepas word te kwantifiseer, soos byvoorbeeld om al drie aspekte gelyktydig in ag te neem. Verder het hierdie studie ook 'n doelwit om te bepaal of 'n noemenswaardige gewigsbesparing gemaak kan word indien die voorgeskrewe verplasingslimiet van 'n raamstruktuur, wat se verplasing as nie-krities beskou kan word soos byvoorbeeld 'n landelike pakhuis, aangepas word.

Hierdie doelwitte word behaal deur beide algehele en strukturele optimering na te vors as ook beskikbare optimeringsalgoritmes. Sagteware is ontwikkel om oplossings vir beide enkel en veeldoelige optimeringsprobleme te vind. Hierdie sagteware word gebruik om verskeie vakwerk probleme vanuit die literatuur te optimeer deur verkillende kombinasies van die voorafgenoemde strukturele aspekte te oorweeg. Die sagteware word ook gebruik om 'n seleksie raamstrukture te optimeer

deur beide gewig en verplasing op 'n veeldoelige wyse te minimeer.

Dit word gevind dat 'n 22% meer ekonomiese oplossing verkry kan word deur al drie die aspekte gelyktydig te beskou teenoor slegs die element groottes. Vanaf die geoptimeerde raamstrukture word dit gevind dat die meerderheid gewig alreeds bespaar kan word voordat die limiet bereik is. Daarom is 'n aanpassing van die limiet nie nodig nie.

## Acknowledgements

I would like to express my sincere gratitude to the following people and organisations for their support throughout this research.

- My supervisors, Mr. E. van der Klashorst and Dr. G.C. van Rooyen, for all the guidance, support and mentorship throughout this research project.
- The Centre for Development of Steel Structures (CDSS) of Stellenbosch University for their financial support which made this project possible.
- To my parents for their loving support and encouragement.
- To my colleagues and friends who provided valuable feedback during the research project and for all the amusement throughout the past two years.
- Cecile Pienaar for all her love and support throughout this project.
- S.R. Bezuidenhout and S.C. van Nierop for assisting in proofreading and editing this document.

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Opsomming</b>	<b>iv</b>
<b>Acknowledgements</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem statement . . . . .	1
1.2 Research aims and objectives . . . . .	2
1.3 Thesis organisation . . . . .	4
<b>2 Background to formal optimization</b>	<b>6</b>
2.1 What is optimization? . . . . .	6
2.2 Multi-objective optimization . . . . .	9
2.3 Analytical and numerical approaches to optimization . . . . .	11
2.4 Derivative free optimization approaches . . . . .	14
<b>3 Optimization from a structural perspective</b>	<b>18</b>
3.1 General mathematical form . . . . .	20
3.2 Types of structural optimization . . . . .	21
3.2.1 Size optimization . . . . .	21
3.2.2 Topology optimization . . . . .	23
3.2.3 Shape optimization . . . . .	24
<b>4 Optimization algorithms</b>	<b>26</b>
4.1 Genetic algorithms . . . . .	26
4.1.1 Background . . . . .	27
4.1.2 Binary encoded variables . . . . .	30
4.1.3 Real-value variables . . . . .	32
4.1.4 Disadvantages of the genetic algorithm . . . . .	33
4.1.5 Multi-objective adaptation . . . . .	33
4.2 Simulated annealing . . . . .	35
4.2.1 Background . . . . .	35
4.2.2 Procedure of simulated annealing . . . . .	37



---

4.2.3	Disadvantages of simulated annealing . . . . .	40
4.2.4	Multi-objective adaptation . . . . .	40
4.3	Particle swarm optimization . . . . .	40
4.3.1	Background . . . . .	41
4.3.2	Procedure of a particle swarm optimization . . . . .	41
4.3.3	Disadvantages of particle swarm optimization . . . . .	45
4.3.4	Multi-objective adaptation . . . . .	45
4.4	Ant colony optimization . . . . .	46
4.4.1	Background . . . . .	47
4.4.2	Procedure of algorithm . . . . .	50
4.4.3	Disadvantages of ant colony optimization . . . . .	51
4.4.4	Multi-objective adaptation . . . . .	52
<b>5</b>	<b>Software implementation</b>	<b>53</b>
5.1	Finite element analysis module . . . . .	53
5.1.1	Cross-sections . . . . .	55
5.1.2	Elements . . . . .	57
5.1.2.1	Truss element . . . . .	57
5.1.2.2	Frame element . . . . .	59
5.1.3	Loads . . . . .	64
5.1.4	Model . . . . .	66
5.1.4.1	Perform an analysis . . . . .	67
5.1.4.2	Copying a model . . . . .	68
5.1.4.3	Meshing a model . . . . .	70
5.1.5	Reporting an analysis . . . . .	71
5.1.6	A note on units . . . . .	72
5.2	Optimization module . . . . .	73
5.2.1	The MOEA Framework, a brief overview . . . . .	74
5.2.2	Adaptation to structural optimization . . . . .	76
5.2.2.1	Objectives and constraints . . . . .	76
5.2.2.2	Basic Problem definition . . . . .	78
5.2.2.3	Size problem definition . . . . .	80
5.2.2.4	Topology problem definition . . . . .	80
5.2.2.5	Shape problem definition . . . . .	81
5.2.2.6	Combination problem definition . . . . .	82

5.2.3	Reporting an optimization . . . . .	83
5.3	Visualization module . . . . .	84
<b>6</b>	<b>Single-objective truss optimization technique comparative study</b>	<b>87</b>
6.1	10-Bar truss . . . . .	89
6.2	25-Bar truss . . . . .	93
6.3	47-Bar truss . . . . .	98
6.4	72-Bar truss . . . . .	104
6.5	Combining results . . . . .	108
<b>7</b>	<b>Multi-objective quantification study</b>	<b>109</b>
7.1	Automatic design module . . . . .	110
7.2	Formal problem definition . . . . .	111
7.3	Example structures . . . . .	114
7.3.1	Plane 4-storey frame . . . . .	114
7.3.2	Plane portal frame . . . . .	117
7.3.3	5-Bay portal frame . . . . .	120
7.3.4	4-Storey building . . . . .	126
7.3.5	Concluding remarks . . . . .	131
<b>8</b>	<b>Conclusions and recommendations</b>	<b>132</b>
8.1	Research overview . . . . .	132
8.2	Consideration of objectives . . . . .	133
8.3	Findings . . . . .	134
8.3.1	Optimization approach comparison . . . . .	134
8.3.2	Multi-objective study . . . . .	135
8.4	Recommendations for future research . . . . .	136
8.5	Concluding statement . . . . .	137
	<b>References</b>	<b>138</b>
	<b>Appendices</b>	<b>148</b>
<b>A</b>	<b>Additional optimization algorithm information</b>	<b>149</b>
A.1	Micro-genetic algorithm . . . . .	149
A.2	Direct search simulated annealing . . . . .	150
A.3	Improvements on particle swarm optimization . . . . .	153

---

A.4	Improvements made on ant colony optimization . . . . .	154
<b>B</b>	<b>Truss optimization cross-section area list</b>	<b>155</b>
B.1	10-Bar truss . . . . .	155
B.2	25-Bar truss . . . . .	155
B.3	47-Bar truss . . . . .	156
B.4	72-Bar truss . . . . .	156
<b>C</b>	<b>Analysis report example</b>	<b>157</b>
<b>D</b>	<b>Optimization report example</b>	<b>161</b>
<b>E</b>	<b>Design report example</b>	<b>164</b>
<b>F</b>	<b>Multi-objective quantification study selected results</b>	<b>176</b>
F.1	Plane 4-storey frame . . . . .	176
F.2	Plane portal frame . . . . .	176
F.3	5-Bay portal frame . . . . .	177
F.4	4-Storey building . . . . .	177

# List of Figures

1.1	Structure of thesis adopted to address the defined objectives . . . . .	5
2.1	Visual representation of a linear programming problem . . . . .	7
2.2	Visual representation of a non-linear programming problem . . . . .	8
2.3	Pareto optimality concept . . . . .	11
2.4	Function with local minima and maxima (Jamil et al. 2013, test function no. 71) . . . . .	12
2.5	5-Element truss used for size optimization . . . . .	16
3.1	Simple example of a structural engineering problem . . . . .	18
3.2	Size optimization - Thicker lines indicate larger elements . . . . .	22
3.3	Topology optimization . . . . .	23
3.4	Shape optimization . . . . .	24
3.5	Comparison between topology and shape optimization (adapted from Auer (2005)) . . . . .	25
4.1	Sequence of selection, crossover and mutation (Adapted from Turing Finance (2016)) . . . . .	29
4.2	Execution process of a GA with elitism . . . . .	30
4.3	The SA process (Rao 2009, p. 706) . . . . .	39
4.4	The PSO algorithm procedure (McCulloch 2016) . . . . .	44
4.5	$\epsilon$ - dominance concept (Deb, Mohan, et al. 2003) . . . . .	46
4.6	The ACO process (Rao 2009, p. 714) . . . . .	48
5.1	Main FEM components . . . . .	54
5.2	Different cross-section forms . . . . .	55
5.3	UML diagram illustrating cross-section class relation . . . . .	56
5.4	A two-dimensional truss element . . . . .	58
5.5	A three-dimensional truss element . . . . .	58
5.6	A two-dimensional frame element . . . . .	60
5.7	A three-dimensional frame element . . . . .	61
5.8	UML diagram illustrating the loads package . . . . .	66
5.9	Structure to demonstrate analysis report . . . . .	72
5.10	UML diagram illustrating the constraint and objective handlers . . . . .	78

## LIST OF FIGURES

5.11	An example of the visualization module . . . . .	85
6.1	10-Bar truss . . . . .	89
6.2	10-Bar truss simultaneous optimization result . . . . .	91
6.3	Performance of the size and SIM approaches for the 10-Bar truss . . . .	92
6.4	Performance of the TS, STS and TSS approaches for the 10-Bar truss .	92
6.5	25-Bar truss . . . . .	94
6.6	Performance of the size and SIM approaches for the 25-Bar truss . . . .	97
6.7	Performance of the TS, STS and TSS approaches for the 25-Bar truss .	97
6.8	47-Bar truss . . . . .	98
6.9	Performance of the size and SIM approaches for the 47-Bar truss . . . .	102
6.10	Performance of the TS, STS and TSS approaches for the 47-Bar truss .	103
6.11	72-Bar truss . . . . .	104
6.12	Performance of the size and SIM approaches for the 72-Bar truss . . . .	107
6.13	Performance of the TS, STS and TSS approaches for the 72-Bar truss .	107
7.1	Example structure for the design report . . . . .	111
7.2	Definition of optimal solution . . . . .	113
7.3	4-Storey plane frame with loads . . . . .	115
7.4	Resulting pareto front of the two-dimensional 4-storey frame . . . . .	116
7.5	Plane portal frame with loads . . . . .	118
7.6	Resulting pareto front of the two-dimensional portal frame . . . . .	119
7.7	Three-dimensional portal frame . . . . .	121
7.8	The wind direction and induced loads on the three-dimensional portal frame . . . . .	123
7.9	Grouping configuration applied to the three-dimensional portal frame .	124
7.10	Resulting pareto front of the three-dimensional portal frame . . . . .	125
7.11	4-Storey frame layouts . . . . .	126
7.12	Grouping configuration applied to the three-dimensional frame structure	129
7.13	Resulting pareto front of the 4-storey frame . . . . .	130
A.1	The DSA process (Sonmez 2007) . . . . .	152

# List of Tables

6.1	Parameters used for the GA . . . . .	88
6.2	10-Bar truss design parameters . . . . .	89
6.3	10-Bar truss results . . . . .	90
6.4	25-Bar truss nodal coordinates . . . . .	94
6.5	25-Bar truss element information . . . . .	95
6.6	25-Bar truss loading information . . . . .	95
6.7	25-Bar truss design parameters . . . . .	95
6.8	25-Bar truss variable detail . . . . .	96
6.9	25-Bar truss results . . . . .	96
6.10	47-Bar truss element definition . . . . .	99
6.11	47-Bar truss design parameters . . . . .	99
6.12	47-Bar truss loading conditions . . . . .	100
6.13	47-Bar truss variable detail . . . . .	101
6.14	47-Bar truss results . . . . .	102
6.15	72-Bar truss design parameter . . . . .	105
6.16	72-Bar truss grouping . . . . .	105
6.17	72-Bar truss loading conditions . . . . .	106
6.18	72-Bar truss results . . . . .	106
7.1	4-Storey frame loading magnitudes . . . . .	115
7.2	Plane portal frame loading magnitudes . . . . .	118
7.3	Three-dimensional frame loading magnitudes . . . . .	128
7.4	Grouping and section assignments for the three-dimensional 4-storey frame . . . . .	128
B.1	10-Bar truss cross-section area list . . . . .	155
B.2	25-Bar truss cross-section area list . . . . .	155
B.3	47-Bar truss cross-section area list . . . . .	156
B.4	72-Bar truss cross-section area list . . . . .	156
F.1	Resulting sections of the plane 4-storey frame . . . . .	176
F.2	Resulting sections of the plane portal frame . . . . .	176
F.3	Resulting sections of the three-dimensional portal frame . . . . .	177

**LIST OF TABLES**

---

F.4	Resulting sections of the three-dimensional 4-storey frame . . . . .	177
-----	--	-----

# Nomenclature

## Abbreviations

$\mu$ GA	Micro Genetic Algorithm
ACO	Ant Colony Optimization
DSA	Direct search simulated annealing
EA	Evolutionary Algorithm
FEA	Finite Element Analysis
FEM	Finite Element Method
GA	Genetic Algorithm
MOEA	Multi-Objective Evolutionary Algorithm
MOOP	Multi-Objective Optimization Problem
MOPSO	Multi-Objective Particle Swarm Optimizer
NSGA-II	Non-Dominated Sorting Genetic Algorithm
PSACO	Pareto Strength Ant Colony Optimization
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SGA	Simple/Standard Genetic Algorithm
SIM	Simultaneous size, shape and topology optimization
SI	International System of Units
SLS	Serviceability Limit State
STS	Size-Topology-Shape optimization
TSS	Topology-Shape-Size optimization



TS	Topology-Size optimization
ULS	Ultimate Limit State
UML	Unified Modelling Language

**Greek Symbols**

$\alpha$	Weight of an objective
----------	------------------------

**Other Symbols**

$f$	Objective function
$\{x\}$	The vector $x$

# 1. Introduction

## 1.1 Problem statement

Optimization, as described by Rao (2009, p. 1), is known as finding the best solution under given conditions. This field has already been studied for a number of years and many strategies have been developed and tested for the solution to specific problems. Due to the magnitude and complexity associated with optimization problems, it has become infeasible to find reasonable solutions without the aid of a computer. Therefore, many software applications have been developed to solve these problems. By doing so, the time required to effectively solve such problems has been reduced while the complexity of the problems that can be solved has increased dramatically.

Optimization can be applied to a number of fields in both research and practice. These applications include engineering, logistical planning, scheduling, reliability, network configurations and many more. Various methods for solving optimization problems have been developed by a number of researchers. A few of these methods are listed below (Chong et al. 2013):

1. Linear programming
2. Newton's method
3. Particle swarm optimization
4. Genetic algorithms

The majority of these methods have proven to be useful in the case of optimizing structures and a vast amount of research has been done in this regard. The sole focus has mainly been on the choice of cross-sections, which is referred to as size optimization. There are however two additional aspects of a structure that can be optimized namely, shape and topology. The shape of a structure refers to the geometric layout and the topology refers to the interconnectivity of members within a structure.

It is notable that some structures can not be optimized with respect to shape and topology. This is due to these specifications being predetermined by other aspects of

---

## 1.2 Research aims and objectives

---

the structure's design. However, there are cases where meaningful cost savings can be achieved by considering all three optimization aspects.

Another influence on the resulting design of a structure is the limit placed on allowable deflection by design codes. By considering both deflection and cost, the design has two conflicting objectives. This means that in the attempt to minimize one objective, the other must be increased. For example, a structure requires more material to reduce its deflection while the use of more material increases the cost of the structure.

It can be argued that for structures situated in areas where deflection will not have a significant influence on its performance, for example rural warehouses and other storage facilities, the deflection limit prescribed by design codes may be adjusted in favour of a significant cost reduction.

## 1.2 Research aims and objectives

Two aims are identified for this study. The first is to investigate the improvement in the structure by optimizing the three aspects of size, shape and topology, simultaneously as opposed to only optimizing the member size of the structure. This will be done by comparing the percentages of reduced weight between the structures resulting from different optimization routines and a base structure which also satisfies the constraints of the optimization problem.

The second aim of this study entails the investigation of the cost that can be saved by increasing the allowable deflection limit for a structure. This only applies when deflection can be regarded as a non-critical aspect, for example, in rural structures and storage facilities.

In order to successfully achieve the aims of this study, the research objectives are broadly defined as:

1. Conduct a literature review on the subject of general and structural optimization. Methods for finding solutions to single- and multi-objective optimization problems and how structural optimization problems can be defined

## 1.2 Research aims and objectives

---

to suit these methods should be considered. This will provide sufficient background for the implementation of a framework that can be used to compare various optimization methods.

2. Consider available meta-heuristic optimization algorithms which can be used to solve optimization problems by means of a computer. It is acknowledged that many algorithms exist and to consider all of them would be infeasible. For the purposes of this study, only one or two of these algorithms should be selected for use in the remainder of the study to optimize various structures.
3. For the determination of displacements and member forces within a structure, a finite element analysis (FEA) module for truss and frame structures needs to be developed in order to execute the optimization routines. These results will be used to determine whether a candidate structure satisfies the constraints of the optimization problem including maximum displacement and member capacity. Allowance for multiple load cases will be advantageous as this reflects a typical design situation and enables using force and deflection values from different load cases for separate calculations, such as for different limit states.
4. The next objective is the development of an optimization framework to solve structural optimization problems for given objectives and constraints. Only truss and frame structures will be considered to match the FEA module. This framework must make allowance for both single- and multi-objective optimization problems in order to accommodate both aims of this research.
5. Results will be obtained by using the developed modules to solve structural optimization problems found in literature. To limit the scope of this study, only truss structures will be considered for this objective. The results from these problems will be used to make a concluding statement for the first aim of this study. Therefore, the structures from these problems must be optimized by considering the size, shape and topology aspects as opposed to only the size aspect of the structure. Considering these aspects individually, sequentially and simultaneously would provide a good comparison to satisfactorily achieve the first aim of this study. Given the variable nature of an optimization routine, the best result from ten consecutive runs of the optimization module will be used as a final result. The results available in literature will be used to validate that the optimization implementation provides reasonable results.

6. In order to achieve the second aim of this study, the final objective is to consider frame structures and optimize them by minimizing the two conflicting objectives of cost and displacement. For these structures, only the member size needs to be optimized as it is assumed that the shape and topology have been prescribed. The allowable limit of deflection for each structure, depicted by design codes, will be used to interpret the optimization results.

## **1.3 Thesis organisation**

The remainder of the thesis is structured as described below and illustrated in figure 1.1:

Firstly, a background section introduces a basic definition of optimization. After which the focus is diverted to the three aspects of structural optimization namely, size, shape and topology.

With an extensive understanding of optimization, some of the available optimization algorithms are discussed. This is required as there is a wide variety of algorithms available, too many to all be used and evaluated. The majority of these algorithms do, however, originate from a few core evolutionary principles.

With the optimization problem defined and the available algorithms examined, the software implementation is discussed. This includes the development of an appropriate FEA and optimization module.

Focus is then placed on the comparison of the results from applying size, shape and topology optimization individually, sequentially and simultaneously to the same structure. Test structures found in literature are evaluated with their respective parameters.

Multi-objective problems are thereafter considered by minimizing the cost, which is quantified by the weight, and displacement of a structure. The result from a multi-objective optimization indicates as to whether or not a significant cost saving can be achieved by slightly increasing the allowable displacement limit of a structure.

### 1.3 Thesis organisation

Four structures are evaluated as test cases.

Finally, conclusions are drawn from the results obtained and recommendations are made for future work to improve and expand the results as well as the tools developed in this study.

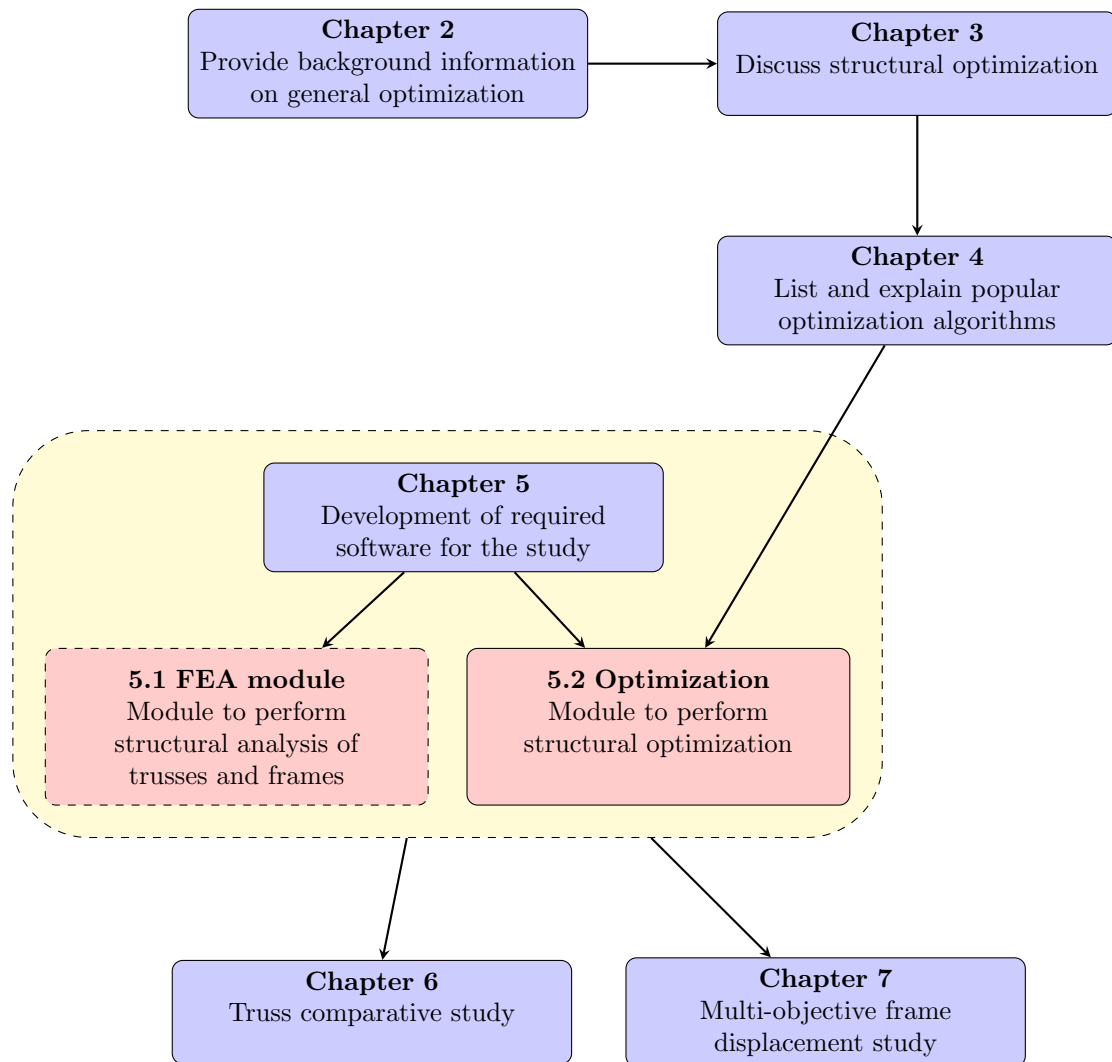


Figure 1.1: Structure of thesis adopted to address the defined objectives

## 2. Background to formal optimization

### 2.1 What is optimization?

According to Arora (2015, p. 1) the definition of optimization is finding the best solution among many feasible solutions. The best solution from a practical point of view can be considered as one which the efficiency of a system is maximized or the cost is minimized. This can in terms of mathematics mean finding the maximum or minimum value of a certain function.

By interpretation of the provided definition, one can realise that almost anything can be optimized to a certain degree. A short example of a linear programming optimization problem can be used to illustrate the concept. The example was adapted from Chong et al. (2013, p. 332). The example is as follows.

$$\begin{aligned} &\text{maximize } 3x_1 + 5x_2 && (2.1) \\ &\text{subjected to } x_1 + 5x_2 \leq 40 \\ &2x_1 + x_2 < 20 \\ &x_1 + x_2 \leq 12 \\ &x_1, x_2 \geq 0 \end{aligned}$$

By visually plotting the above problem and its given constraints in figure 2.1, the area of possible solutions is easily identified. The optimization component of this example is searching for and finding the maximum value of the given function. In other words, searching through all possible solutions and determining the best solution for the given constraints.

## 2.1 What is optimization?

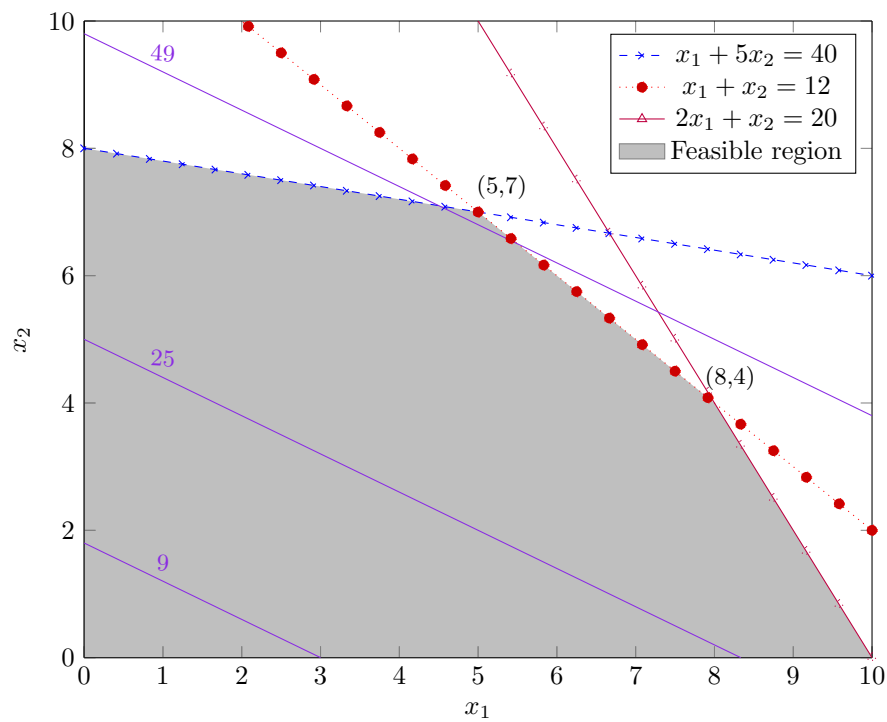


Figure 2.1: Visual representation of a linear programming problem

By examining the graphical solution, points  $(5,7)$ ,  $(8,4)$ ,  $(0,8)$  and  $(10,0)$  can be identified as possible points where the function can be at its maximum. These points are considered the corners of the search space. With the assistance of the contours the optimal solution can be identified as  $(5,7)$ .

In this example the objective was to maximize the value of  $3x_1 + 5x_2$  within the provided constraints. This is the fundamental concept of an optimization problem, having to either maximize or minimize a certain function, or combination of functions, subjected to some constraints.

It is also important to realise that, when the complexity of the problem constraints or the so-called “objective function” increases, the problem can become extremely complex. Referring to figure 2.1, the addition of more variables will increase the dimension of the problem and the region of feasible solutions may take an irregular shape.

To illustrate this statement, a few minor adjustments can be made to the problem of equation 2.1 to make the problem non-linear. Consider the adjusted problem as in



## 2.1 What is optimization?

equation 2.2.

$$\begin{aligned}
 &\text{maximize } 3x_1^2 + 5x_2^2 && (2.2) \\
 &\text{subjected to } x_1^2 + 9x_2 \leq 90 \\
 &2x_1 + x_2 < 19 \\
 &x_1 + x_2 \leq 12 \\
 &x_1, x_2 \geq 0
 \end{aligned}$$

The first noticeable differences between the two problems is that the objective function now takes the form of a circle and one constraint is a parabola. A problem of this nature is usually termed a non-linear programming problem. The modified problem can, same as before, be graphically plotted as in figure 2.2.

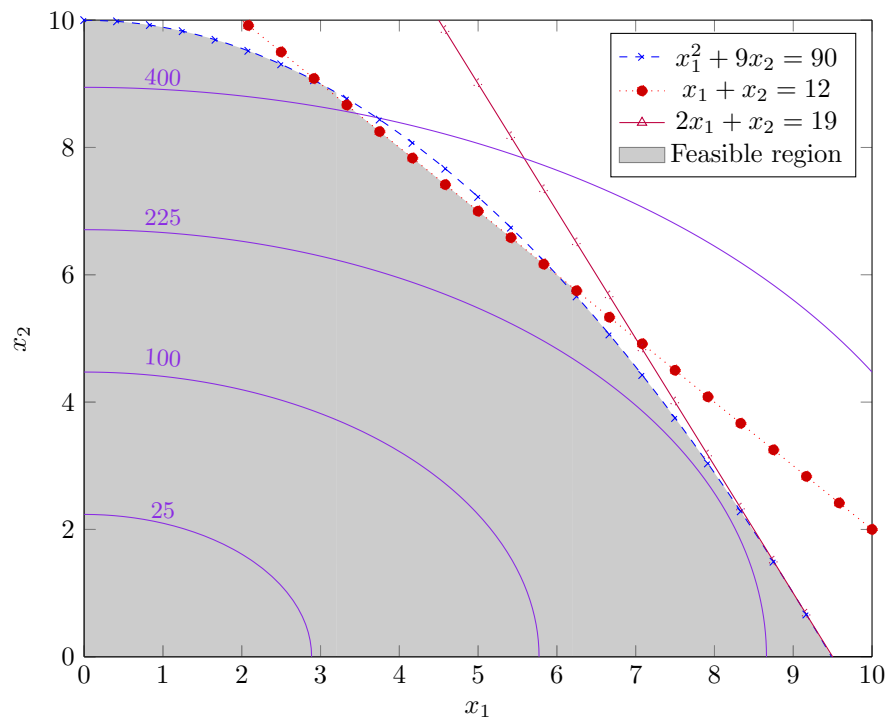


Figure 2.2: Visual representation of a non-linear programming problem

The solution to the modified problem is significantly more difficult to obtain. In the previous problem the solution can be deduced from the graphical representation. This is no longer possible since there are no longer specific corners that can be identified as points of interest, but rather entire edges. Therefore, the solution must be obtained through a more advanced approach.

---

## 2.2 Multi-objective optimization

---

Being able to move from a trivial problem to a more complicated problem by making small modifications proves the computational complexity associated with optimization problems. The constraints, objective function and the number of variables present in the problem can have a big influence on its complexity. Typically a problem with more variables is considered to be more complex.

According to Arora (2015, p. 3), the aforementioned objective function of an optimization problem can be defined as the function that is either maximized or minimized. By evaluating values of the objective function with respect to different solutions, one can compare the fitness of one solution to another. The objective function is used to determine which of the two solutions is classified as the superior solution. It is sometimes also referred to as the criterion or the merit by which solutions are compared.

## 2.2 Multi-objective optimization

In more complex optimization problems, there may be more than one objective function present. These problems are referred to as multi-objective optimization problems (MOOPs). This section describes how these problems are typically addressed.

The objectives of multi-objective problems are often conflicting. For example in ship design where efficiency would be improved by minimizing the required engine power, but doing so conflicts with safety objectives which require reserve capacity (Korpus 2015). Rao (2009, p. 9) mentions that in structural design the minimum weight design does not always correspond to the minimum stress or lowest cost design. The selection of the objective function directly influences the solution to the problem. Therefore, one of the most important choices in an optimization problem is the choice of the objective function (Rao 2009, p. 9).

There are generally two well-known methods for dealing with MOOPs. The first and simplest approach to solving this problem is by reducing a multi-objective problem to a single-objective problem by using the weighted sum method (Deb 2001, p. 48). This method uses a linear combination of the objective functions as one objective, where

## 2.2 Multi-objective optimization

---

each of the objective functions are assigned a weight,  $\alpha_i$ . This can be expressed as in equation 2.3.

$$f(X) = \alpha_1 f_1(X) + \alpha_2 f_2(X) \quad (2.3)$$

Where  $f(X)$  denotes an objective function and  $\alpha$  denotes the relative importance or weight of one objective function to another.

One drawback with this approach is that it only yields a single solution as a result of the problem being reduced to one of a single-objective nature. In reality only MOOPs for which the objectives are not conflicting have a solution where all the objectives are at their optimum (Miettinen 1998, p. 5).

For the majority of MOOPs with conflicting objectives, it is impossible to find a single solution at which all the objectives are at their optimum (Miettinen 1998, p. 11). For this reason, the concept of pareto optimality was introduced. A solution is regarded as pareto optimal when none of the objective values can be improved without compromising one of the other objective values (Hwang et al. 1979, p. 16). In some texts, a pareto optimal solution is also called a non-dominated solution.

To illustrate the concept of pareto optimality, consider figure 2.3. In this figure a set of possible solutions is plotted according to their objective values. The solutions which can be considered as pareto solutions are highlighted. The line formed by all the pareto solutions is called the pareto front. Only two objectives are used in this figure for simplicity, but this number may be increased depending on the nature of the problem.

## 2.3 Analytical and numerical approaches to optimization

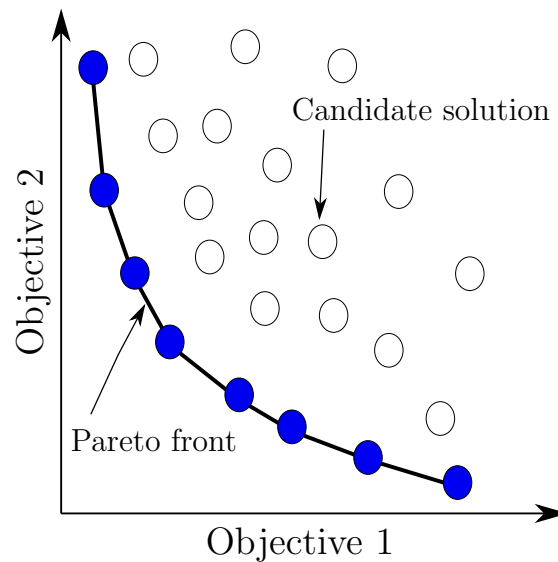


Figure 2.3: Pareto optimality concept

By considering each objective individually, this approach to a MOOP produces a number of possible solutions as a result. The best solution can then be chosen based on other limits or considerations not included in the optimization such as a desirable cost or some other criteria.

In summary, optimization is the selection of the best fit solution out of many solutions. This is done by using one or more objective functions which is able to quantify the fitness of a solution and comparing it to the fitness of other possible solutions. Special consideration is given for dealing with problems containing multiple objectives, considering that they may have multiple solutions. In the next section the approaches to solving optimization problems are discussed.

## 2.3 Analytical and numerical approaches to optimization

Finding the solution to an optimization problem can be relatively easy if the problem is of simple nature (Rothlauf 2011, p. 46). This means that if the problem is well defined, with little to no constraints, a solution can be found with minimal effort. For this discussion, only single-objective problems are considered, but the concepts

## 2.3 Analytical and numerical approaches to optimization

can be adapted to suit multi-objective problems.

According to Rothlauf (2011, p. 46), simple problems can be solved in the manner of determining the points where the value of the function's gradient is zero. For the case where the function,  $f(x)$ , has multiple variables, a vector containing the partial derivatives can be considered. This is illustrated by equation 2.4.

$$\nabla f(x) = \left( \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)^T \quad (2.4)$$

After determining all the so-called stationary points where  $\nabla(f(x)) = 0$ , it can be determined which of these points are classified as local maxima or minima. Figure 2.4 shows an example of a function with multiple local minima and maxima.

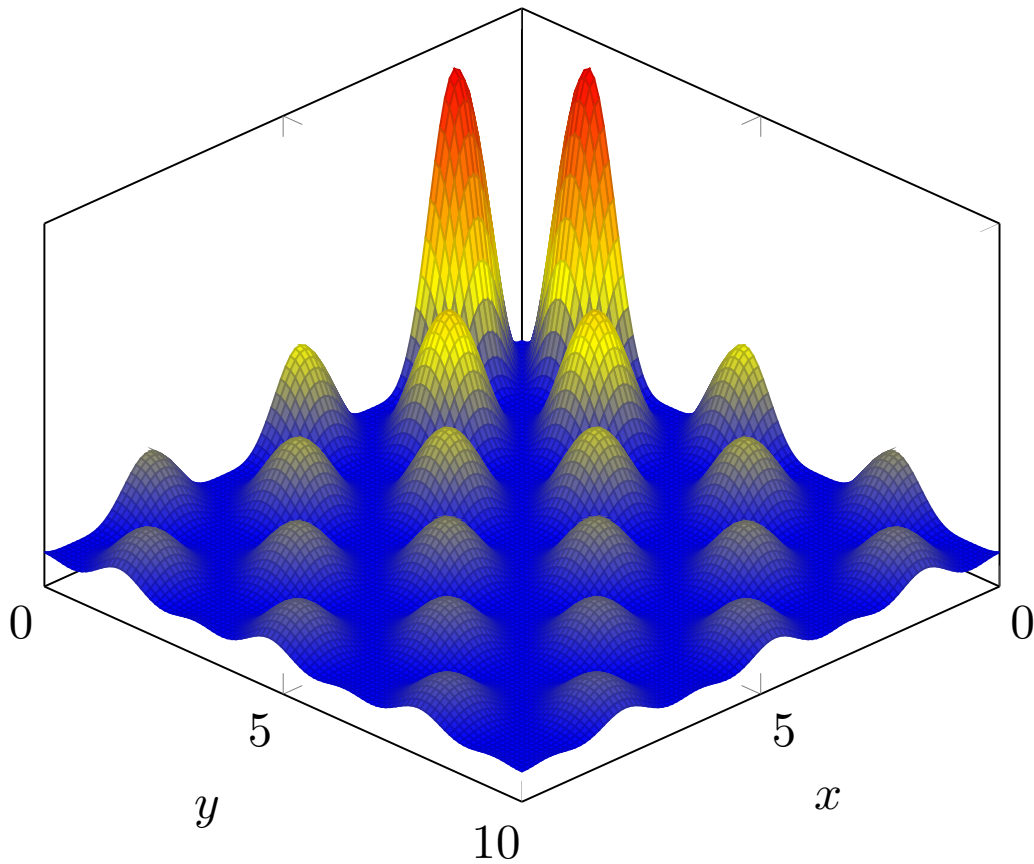


Figure 2.4: Function with local minima and maxima (Jamil et al. 2013, test function no. 71)

Rothlauf (2011, p. 47) mentions that a simple method of distinguishing between local minima or maxima is by calculating two function values adjacent to the stationary

## 2.3 Analytical and numerical approaches to optimization

point. By analysing these values, it can be stated a point is a local minima if the function values at the points in close proximity to it are larger than at the point itself. The opposite applies to local maxima.

An equivalent approach is to determine the Hessian matrix of the objective function. This can be done as shown in equation 2.5 (Rothlauf 2011, p. 47).

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (2.5)$$

If the determinant of the matrix,  $H(f)$ , is equal to zero and this matrix is positive definite at a stationary point, then the point is a local minimum. On the other hand, in the case where the matrix is negative definite, the point is a local maximum.

In the case where constraints are applied to the problem, there are cut-off or boundary points where the function values beyond these points are not considered when searching for local maxima or minima. These points must also be considered as stationary points during the identification of local maxima and minima. The reason for this is that those boundary points may not be stationary points if the function is considered in its entirety, but if the search is ended at these points, they might be points of local minima or maxima for the specific problem at hand.

The analytical approach described above is one of many similar mathematical approaches. There are numerous literature on the full mathematical derivation of these methods and the variations thereof. Lange (2013) provides a good in-depth explanation of some of these methods, and is summarized below.

### 1. MM algorithm

An iterative method useful for high-dimensional problems such as image reconstruction.

## 2.4 Derivative free optimization approaches

---

### 2. EM algorithm

A special case of the MM algorithm which is able to function without a complete set of data by reconstructing the data from the input parameters.

### 3. Newton's method

A very popular method for low dimensional problems, it uses the gradient of the objective function to steer the search towards a better solution.

### 4. MM gradient

A combination of the MM algorithm and Newton's method. The optimization step of the MM algorithm is solved by using Newton's method.

### 5. Conjugate gradient

A method suited for high-dimensional problems that do not rely on the second derivative of the objective function or the inversion of matrices. It also requires exact line searches.

### 6. Quasi-Newton

Very similar to the conjugate gradient methods, but it does rely on the inversion of matrices and operates on inexact line searches.

These methods are typically not used for practical engineering problems because they are intended for ideal problems. Rothlauf (2011, p. 45) notes that derivative based methods are preferred for problems where the effort grows polynomially with the problem size, this is generally not the case for engineering problems. For engineering optimization problems, the effort required by derivative based methods is immense.

For the apparent inapplicability of derivative based methods, a detailed analysis and discussion of these methods would be unnecessary. The research needs to be shifted into optimization approaches which are applicable to real world scenarios, such as meta-heuristic methods. This is done in the upcoming section.

## 2.4 Derivative free optimization approaches

For the majority of mathematical optimization techniques, an optimum solution is found by utilising the derivative of the objective function. However, Conn et al. (2009, p. 1) notes that there are cases when it is not possible to obtain such a

## 2.4 Derivative free optimization approaches

---

derivative, but the optimization must still be completed. Goldberg et al. (1989, p. 3) mention that for a practical implementation of optimization, a derivative based approach only caters for a small number of problems because the real world is full of discontinuities and huge search spaces.

Conn et al. (2009, p. 2) explain that, as the scale and complexity of an optimization problem increase, more sophisticated derivative-based optimization methods become essential to solving large scale problems. This requires the user of the software implementation to provide the derivative to the software in order to perform the optimization routine. Furthermore, Goldberg et al. (1989, p. 3) note that the numerical approximation of derivatives also has its shortcomings. These approximations tend to be both inaccurate and impractical as it could lead to a noticeable increase in the computational time required to obtain a result.

With regards to developing an optimization module for structural optimization, as required for this study, it is possible to consider the actual optimization routine as a so-called *black box* operation. This implies that the optimization routine is independent of the given application. This is useful in terms of simplifying the extension of an existing application with the aforementioned module. A disadvantage of such a design is that the actual objective function used by the module differs for each application. Therefore, the use of an approach where the derivative of the objective function is required is deemed inapplicable for inclusion in the development of the optimization module for this study.

For these reasons, derivative free methods seem to be a better approach for the solution of optimization problems when compared to the traditional derivative-based methods. Researchers developed a number of derivative free optimization approaches. A few of which are mentioned by Rios et al. (2013) are listed below.

### 1. Local search methods

- Nelder-Mead simplex
- Generalized pattern search
- Generating set search
- Trust-region methods
- Implicit filtering



## 2.4 Derivative free optimization approaches

### 2. Global search methods

- Lipschitzian-based partitioning
- Multilevel coordinate search
- Response surface
- Surrogate management framework
- Branch-and-fit
- Hit-and-run
- Particle swarm algorithms
- Genetic algorithms
- Simulated annealing
- Ant colony optimization

The concept behind derivative free optimization techniques is elementary. Instead of relying on derivative information, approximated or exactly determined, from the objective function, it focusses on locating an optimal or near optimal solution by utilising a sample set of function values (Conn et al. 2009, p. 2). This implies that derivative free methods compare various collections of possible solutions to the optimization problem. This comparison is performed to determine which characteristics of the available solutions are better suited for the objective function. By utilising the information of which solution characteristics produce better results, the overall solution of the problem can be improved. New solutions are normally generated from this information and are also compared to the previous collection of solutions. This procedure is usually repeated a number of times until a solution of a certain degree of *fitness* or *goodness* is obtained.

The search space size of an optimization problem varies for every problem. It can either be a discrete space with a finite number of solutions, or a continuous one with an infinite amount of possible solutions (Christensen et al. 2008, p. 7). The number of solutions in a discrete space can vary, but the number of solutions associated with an engineering problem is usually large. To illustrate this, consider a size optimization of a 5-element truss as in figure 2.5.

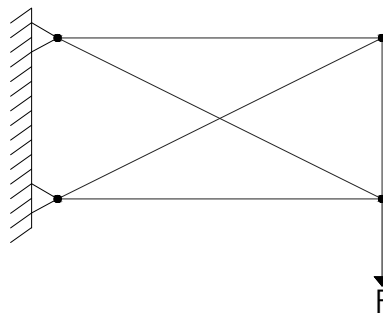


Figure 2.5: 5-Element truss used for size optimization

## 2.4 Derivative free optimization approaches

---

Presume that only equal-leg angle-sections may be used for the given structure, then according to the Southern African Institute of Steel Construction (2013) there are 46 available sections. For the 5 members,  $46^5$  possible solutions exist for this small structure. This number would increase significantly if other sections are also considered. This example shows that the search space can be large, even for small problems.

Since derivative free methods follow an iterative process of obtaining, evaluating and selecting the best solutions, they can easily accumulate a large amount of data that must be processed. This sizeable amount of data makes these methods suitable for using the aid of a computer to obtain an answer. However, in some cases the computational requirement could be so much that a computer program still requires a substantial amount of execution time in order to reach a satisfactory solution to the optimization problem. Due to this predicament, it is recommended that, if possible, a parallel computing approach is used (Conn et al. 2009, p. 6). By doing so, the execution time of such a program may be significantly reduced, which would be an important advantage with respect to typical time constraints.

One should be aware that there are significant drawbacks to not having the derivative information. The lack thereof influences many aspects of solving an optimization problem including the scale of the problem, stopping criteria and accuracy of the solution (Conn et al. 2009, p. 2). Although derivative free methods are not superior to derivative based methods, for the solution of current real world optimization problems the known limitations of derivative free methods are deemed acceptable. Furthermore, the current dilemmas associated in determining the derivative of complex objective functions serve as the main motivation for the usage of derivative free methods.

### 3. Optimization from a structural perspective

Due to an increasing scarcity of structural materials and environmental considerations, the need for more economical structures has substantially increased in recent years (Kirsch 1993, p. 1). The term economical refers to lighter and more cost-effective structures. This chapter describes how structural optimization is generally defined to match formal optimization definitions.

Christensen et al. (2008, p. 1) define structural optimization as: “The subject of making an assemblage of materials to sustain loads in the best way.” In this definition an assemblage of materials that sustains load can be considered as a structure (Gordon 1978). As a conceptual example, adapted from Christensen et al. (2008, p. 1), consider a load at a certain position that has to be transferred to a support at another position. The situation is illustrated in figure 3.1.

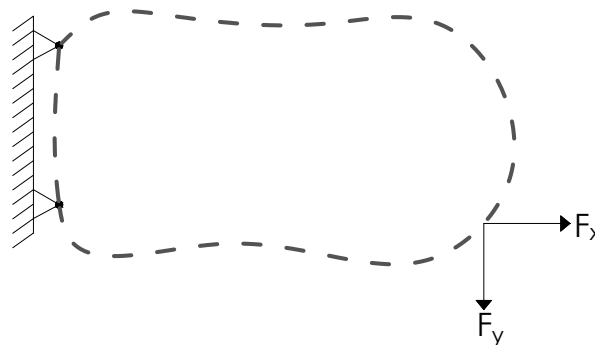


Figure 3.1: Simple example of a structural engineering problem

It is the structural engineer’s responsibility to design a structure that can successfully transfer the applied load to the supports, without the structure collapsing. It is obvious that this problem may have numerous solutions. For this reason, it can be considered as an optimization problem. The objective in this case could be to find a structure that transfers the load in the best possible way.

---

For structural optimization problems, one must raise the question as to what is considered the “best” or “optimal” structure. This is usually a matter of perspective. One might view the optimal structure as one with minimal weight, which has an influence on the cost of the structure. Another view might be to make the structure as stiff as possible or to be as resistant to instability as possible (Christensen et al. 2008, p. 1). In industry, the optimal structure may be defined as one which satisfies design code requirements for the least cost. There are many more considerations and it could even be a combination of these considerations. All of these considerations may be termed as objectives to the optimization problem.

The above-mentioned objectives are considered mathematically so that they can be expressed in terms of values that a computer then uses to compare solutions. There are also non-mathematical factors to be considered such as functionality, economy and aesthetics (Christensen et al. 2008, p. 1). These factors are quite difficult to express mathematically for use in optimization problems and even more so for computer programs as it would make such an application quite complicated. For the purpose of this research these subjective factors are not taken into consideration.

All these considerations must also have limits. Otherwise the result may be a structure which is designed to withstand more loading than it will be subjected to, but it meets the criterion for the optimal solution to this problem perfectly. To avoid results of this nature, a combination of considerations can be used. For example, a structure has to be as stiff as possible to resist the load but also have a minimum weight. These criteria can be applied as a linear combination of objectives as illustrated in equation 2.3. The maximum and minimum bounds of such properties of the structure can also be added to the problem as constraints. By doing so the search space of the problem is reduced, potentially decreasing the time required to obtain a feasible solution.

The time available to a structural engineer for the successful design and optimization of a structure is usually limited. This is due to increasing pressure to meet project deadlines and avoid additional costs or other penalties. For this reason, the time consumed by the optimization of a structure cannot be excessive. Therefore, the optimization process itself must be as time efficient as possible in order to make it usable for engineers. This serves as motivation to make simplifications to the

### 3.1 General mathematical form

optimization process which improves efficiency, but these simplifications typically compromise the accuracy of the results to a certain degree.

## 3.1 General mathematical form

The mathematical form of structural optimization looks similar to equation 2.1 and can be described as follows (Christensen et al. 2008, p. 3):

$$\begin{aligned} &\text{minimize } f(x_1, \dots, x_n, y_1, \dots, y_n) = f(\{x\}, \{y\}) \\ &\text{subjected to: Behavioural constraints on } y_i \\ &\quad \text{Design constraints on } x_i \\ &\quad \text{Equilibrium constraints} \end{aligned} \tag{3.1}$$

Where:

1.  $f(x_1, \dots, x_n, y_1, \dots, y_n)$  denotes the objective function. For example, the weight or cost of the structure.
2.  $x_i$  denotes the design variable. This describes the design of the structure i.e. the member configuration and properties. It is useful to write all these variables in vector form as,  $\{x\}$ .
3.  $y_i$  denotes the state variable. This refers to the response of the structure i.e. the displacements obtained by performing a Finite Element Analysis (FEA). It can also be written in vector form as,  $\{y\}$ .

The case may arise where more than one objective function is considered, resulting in a multi-objective problem. This problem may be addressed in either of the two ways presented in section 2.2. The first being a linear combination of the objectives expressed in equation 2.3, rewritten for  $n$  objectives shown in equation 3.2. It is important to note that the sum of all the weights must result in a value of 1, mathematically expressed in equation 3.3.

$$f(\{x\}, \{y\}) = \alpha_1 f_1(\{x\}, \{y\}) + \alpha_2 f_2(\{x\}, \{y\}) + \dots + \alpha_n f_n(\{x\}, \{y\}) \tag{3.2}$$

$$\sum_{i=1}^n \alpha_i = 1 \tag{3.3}$$

---

## 3.2 Types of structural optimization

---

For the case where a weighted objective function is used, one typically tries to achieve a pareto solution (Christensen et al. 2008, p. 3). The position of this solution on the pareto front will depend on the chosen weights assigned to the problem. Therefore, the values of these weights should be chosen after careful consideration. A possible strategy is to obtain solutions corresponding to different choices of weights and comparing them. Doing so will provide an understanding of how certain weight distributions influence the final solution. From this information, the structural engineer can decide on an appropriate choice for the values of  $\alpha_i$ . However, following this strategy can be very inefficient with respect to the time required to solve the optimization problem for various weight distributions. It is especially true for large problems that consume a substantial amount of time to arrive at just one solution.

Alternatively, it would be more time efficient to utilise the multi-objective approach which reveals a pareto set of solutions as illustrated in figure 2.3. This would require the optimization to run only once and the structural engineer is then able to decide which one of the solutions on the pareto front best meets the design requirements.

## 3.2 Types of structural optimization

There are typically three types of structural optimization, namely topology, size and shape optimization. These types are discussed in sections 3.2.1 to 3.2.3.

### 3.2.1 Size optimization

Christensen et al. (2008, p. 5) define size optimization of structures as the optimization of the size of elements in the structure, while the geometry and element connectivity remains unaltered. This may be element thickness or cross-sectional area, and is typically used for truss and frame structures. The concept of size optimization is illustrated in figure 3.2.

### 3.2 Types of structural optimization

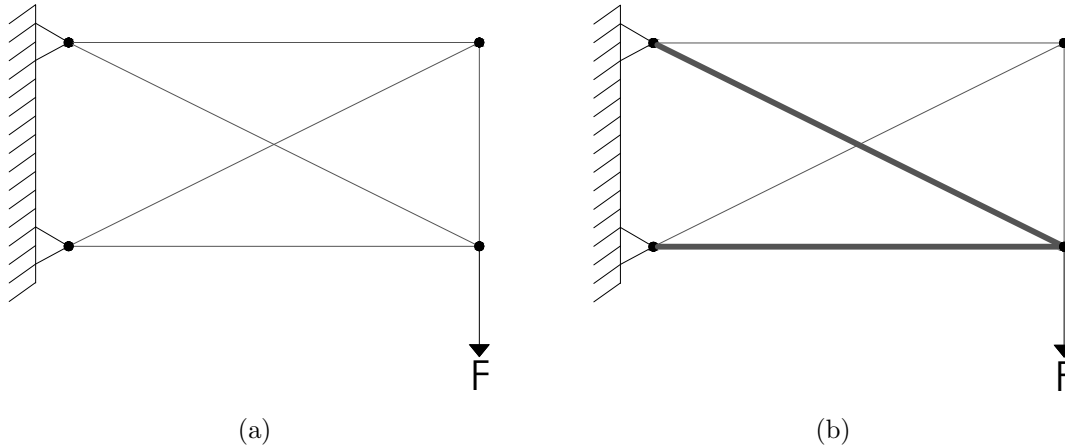


Figure 3.2: Size optimization - Thicker lines indicate larger elements

Size optimization can typically be defined as cross-section or member selection. The size of the member is typically determined by its cost, which can be quantified by its weight or availability, and its ability to meet the design requirements. In several structures, additional constraints are also placed on the choice of the members, for example only a certain type of cross-section such as I- or H-profiles.

It is important to note that the use of a large number of different member sizes can have a significant impact on the fabrication costs of the structure. For example, a light structure comprising of many different cross-sections is inherently more difficult to construct and may be prone to construction errors such as placing a member at an incorrect position. Furthermore, the use of a variety of different cross-section may increase the fabrication and transport costs.

Various research publications have applied size optimization. For example, Barraza et al. (2017) optimized frames for seismic loads while many other publications focused on developing an efficient algorithm for performing size optimization. These include Degertekin (2013), Kaveh and Talatahari (2009b), and Gonçalves et al. (2015) which optimised the size of various structures using teaching-learning-based optimization, the big bang–big crunch algorithm and the search group algorithm respectively.

## 3.2 Types of structural optimization

### 3.2.2 Topology optimization

Auer (2005) states that topology optimization refers to the element-node connectivity within a structure. In the context of truss and frame structures, this can be attributed to the element configuration of the structure with respect to the predefined nodes. Topology optimization is also the most general form of structural optimization (Christensen et al. 2008, p. 5).

In the case of a truss or a frame structure as in figure 3.3, topology optimization is performed by connecting all the nodes with structural elements as shown in figure 3.3a. Elements are then allowed to be removable during the optimization routine. By doing so all the initially defined excess elements are removed and the solution is the remaining elements as in figure 3.3b.

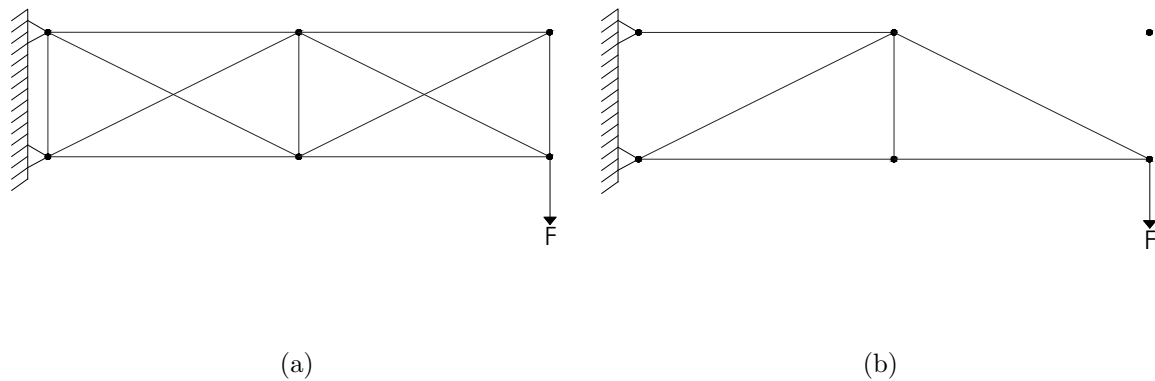


Figure 3.3: Topology optimization

Topology optimization has been employed by a number of research papers. Goo et al. (2016) optimized the topology of thin plate structures while Xia et al. (2013) presented a method for optimizing the topology of a structure. Other studies considered topology with size and shape optimization to improve the optimization results and test their proposed optimization algorithms (Achtziger 2007; Ahrari et al. 2015; Miguel et al. 2013).



## 3.2 Types of structural optimization

### 3.2.3 Shape optimization

Shape optimization relates to the contour or domain of a structure (Christensen et al. 2008, p. 5). This can be related to the physical shape of elements in a structure. Typically, a solid element in a structure is chosen to be of either rectangular or circular form. This doesn't need to be the case in a shape optimized structure. Consider a rectangular beam as in figure 3.4a of a constant width. To reduce the amount of material required, the shape of the beam can be changed as in figure 3.4b, which is still able to successfully transfer the load to the support.

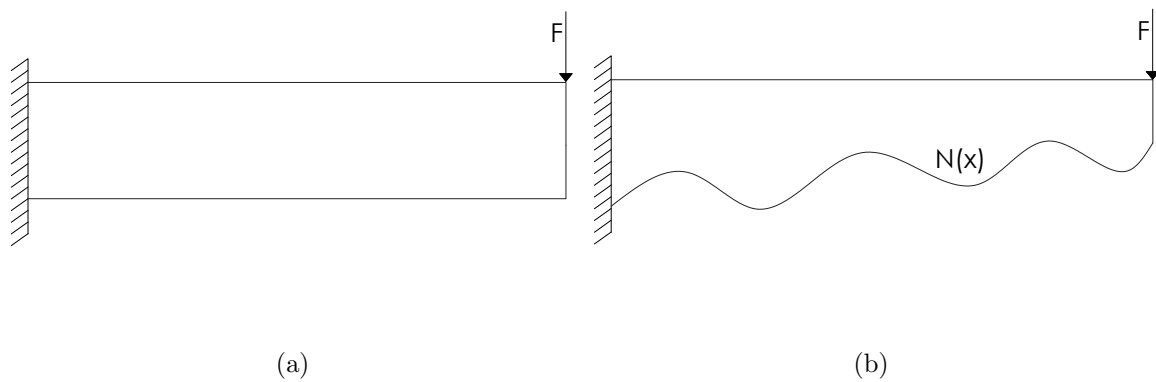
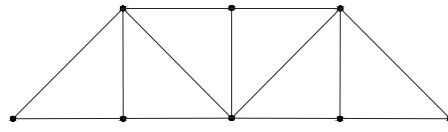


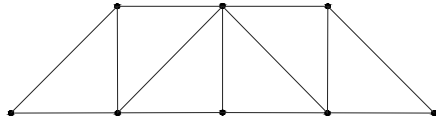
Figure 3.4: Shape optimization

In the case of truss structures, it is important to not confuse shape and topology optimization. Shape optimization does not change the elemental configuration of the structure, only the positioning of nodes in the structure. A comparison is shown in figure 3.5.

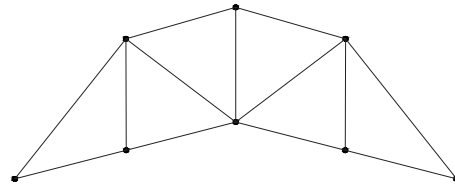
### 3.2 Types of structural optimization



(a) Original



(b) Topology



(c) Shape

Figure 3.5: Comparison between topology and shape optimization (adapted from Auer (2005))

Shape optimization has been used to optimize structures by a number of researchers. For example, Wang et al. (2002) optimized the shape of truss structures under multiple displacements constraints and Nasrollahi (2017) optimized the shape of large span trusses.

It is possible to consider combinations of these optimization types. The combination of all three of these methods is termed “layout” (Auer 2005) or simultaneous optimization. In most implementations, only one or two of these optimization types are typically considered, since these types can sometimes produce contradicting results.

In several structures, it may not be possible to optimize the topology or the shape of the structure due to them being defined by other aspects of the structure’s design phase. For example, the nodal positioning of a structure may be predetermined for aesthetic reasons.

## 4. Optimization algorithms

As previously stated, derivative free methods for solving optimization problems prove to be appropriate for practical problems. Therefore, this chapter is dedicated to the discussion of popular optimization algorithms which use derivative free techniques. From these algorithms, a decision is made on the algorithm best suited to the optimization implementation of this study.

The majority of these algorithms are classified as evolutionary algorithms (EAs). EAs refer to algorithms based on principles found in nature. For example, the characteristics and behaviour of biological and molecular systems. The principles of these algorithms are presented in the following sections with additional information presented in appendix [A](#).

Considering that both single- and multi-objective optimization problems are of interest in this study, the adaptations made to these popular algorithms to cater for multi-objective problems are also discussed. Multi-objective in this case refers to algorithms that consider all the objectives individually and not as a single-objective problem where the objective function comprises of a linear combination of the respective objectives.

### 4.1 Genetic algorithms

Genetic algorithms (GAs) are the earliest and one of the most widely used forms of EAs (Simon [2013](#), p. 35). They are defined by Goldberg et al. ([1989](#), p. 1) as search algorithms based on the mechanics of natural selection and natural genetics. In general, they filter through generations of solutions, where the solutions improve for every generation.

In this section a brief background of the GA is provided. Two different variable types normally used by a GA, namely binary and real-value variables, are discussed as well as how the typical GA operations are defined for each respective variable type. An adaptation to the GA for multi-objective problems is also described in section [4.1.5](#).

### 4.1.1 Background

Since the GA is based on biological processes, it often uses terminology derived from biology. In order to discuss the workings of a GA, one must have an understanding of these terms in the context of a GA. Short descriptions of these terms are listed below as described by Mitchell (1999, p. 5).

1. Chromosome or individual - A candidate solution to a problem, comprising of various genes. For example, a truss structure that satisfies a certain optimization problem.
2. Gene - A single characteristic that describes the chromosome. For example, the area of an element within a truss solution.
3. Population - A collection of individuals for a certain generation. It can be viewed as many trusses that are all candidate solutions to an optimization problem. To start the GA an initial population is randomly generated.
4. Generation - The population used by the algorithm during a certain iteration. Each generation is an improved version of its predecessor. For example, a generation is a collection of trusses that are adapted to find better solutions than themselves.
5. Elitism - This is the retaining of the best-fit solution(s) from one generation to the next (De Jong 1975, p. 101). For example, the five lightest structures from the current generation is directly transferred to the next generation.

The main operators of a GA are selection, crossover and mutation (Coley 1997, p. 10). These operators are used to improve the solutions from generation to generation by means of utilising the genes that produce good solutions from a certain population. They are defined as follows:

1. Selection

On a similar principal as natural selection, selection applies pressure on a population in an attempt to eliminate weak performing individuals. This ensures that fitter individuals have a better probability to pass their genes on to the next generations (Coley 1997, p. 10). This operator promotes the idea that fitter genes are passed to the following generations which increase the

chances of obtaining better solutions.

A popular selection strategy is tournament selection. This strategy selects a random portion of the current population, and the best fit individual is inserted into a mating pool from which the next generation's population is created (Miller et al. 1995). Tournament selection is repeated until enough individuals have been selected to create the next generation.

## 2. Crossover

This operation produces a new individual from two existing individuals (De Jong 1975, p. 21). It is best to view crossover as the scenario of two parents producing a child, where the child's genes are a combination of both of its parents' genes. There can be various versions of children from two single parents, simply on the basis of which genes are obtained from which parent. Therefore, it is possible to produce a number of children from two parents. The idea is that the good genes from both parents are retained and the not-as-good genes are replaced by good genes from the other parent. In theory, this operation will ideally result in a child which is a better individual than each of its parents. An example of crossover is the interchange of members between two structures to produce a new structure. This new structure is a combination of the two and is then placed in the next generation as a candidate solution.

## 3. Mutation

Mutation generates a new individual by means of independently modifying one or more genes of an existing individual (De Jong 1975, p. 22). The specific gene or genes that are mutated are selected randomly with the assistance of a statistical distribution, for example a normal distribution. The probability of an individual mutating, the mutation rate, can be selected before the start of the optimization. It is typically chosen as a small percentage, usually  $< 5\%$ , to avoid excessive mutation. In a structural context, an example of mutation may be randomly changing the size of a member.

These operations are usually performed sequentially during the execution of a GA. The order of the execution is typically selection, followed by crossover which is succeeded by mutation. This sequence is illustrated in figure 4.1, where the respective

## 4.1 Genetic algorithms

colours indicate the genes from a single individual.

A graphical representation of a GA algorithm is shown in figure 4.2. This figure illustrates the procedure of a GA using the elitism strategy. This strategy moves a predefined number of best solutions from one generation to the next. By doing so, it is ensured that potentially good solutions are not lost during the evolutionary process.

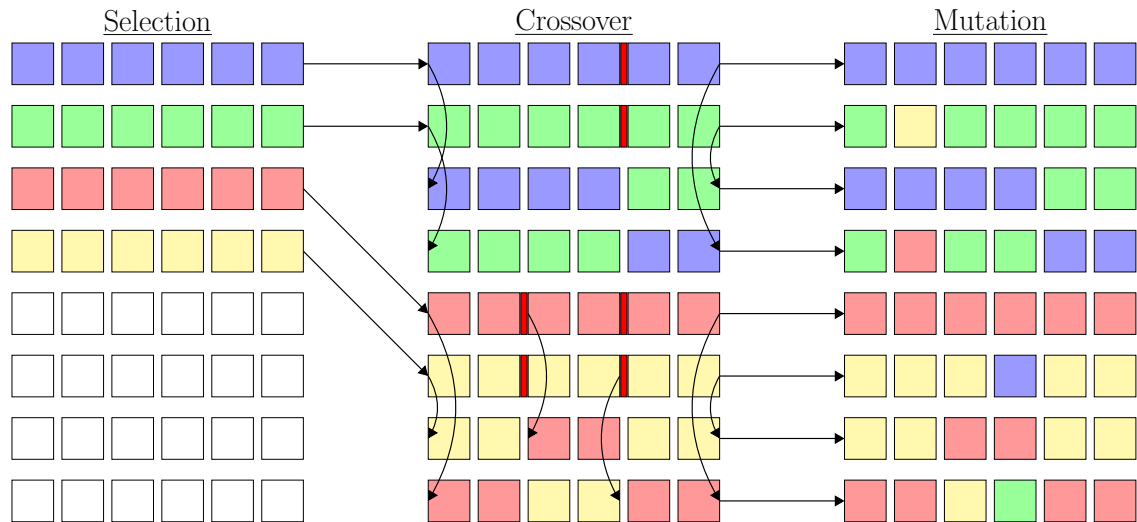


Figure 4.1: Sequence of selection, crossover and mutation (Adapted from Turing Finance (2016))

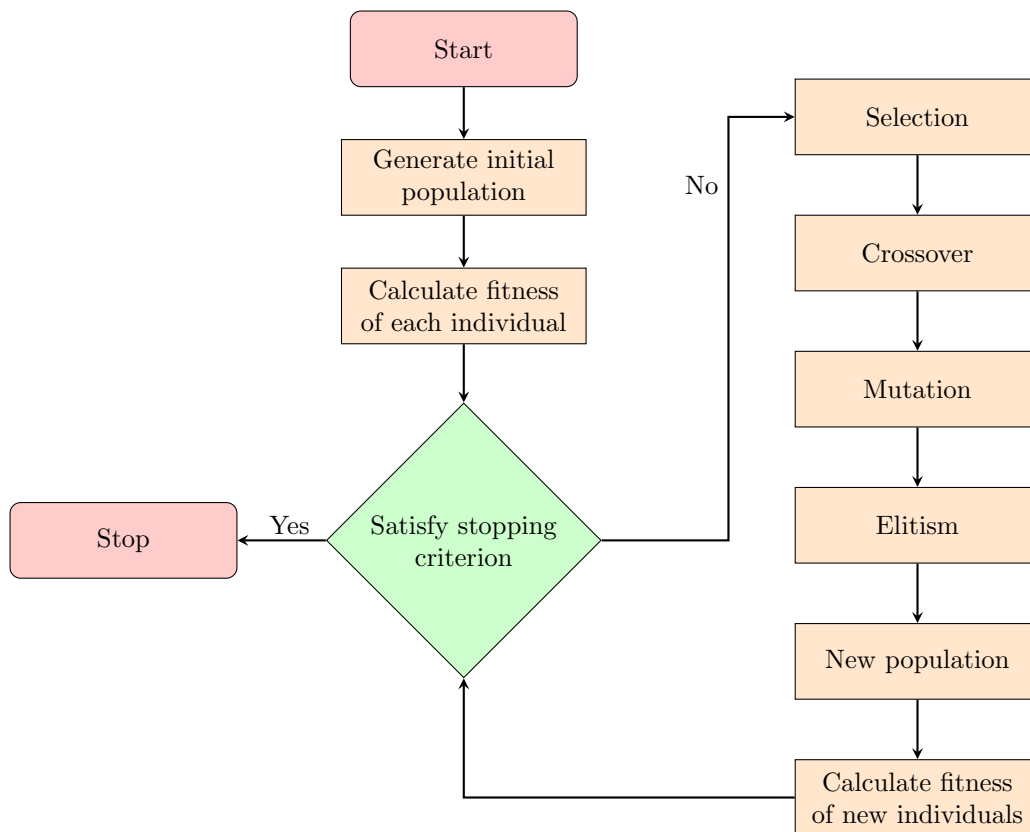


Figure 4.2: Execution process of a GA with elitism

The GA has a number of parameters that must be chosen beforehand. These parameters include the choice of the population size and the detail or criteria on how the selection, crossover and mutation are going to be performed. If the elitism strategy is included in the GA, the number of elite solutions which are transferred from one generation to the next must also be prescribed. In addition, the approach used to generate the initial population and encode the variables of the GA is an important part of the algorithm. A number of these approaches have been developed, each with its own advantages and disadvantages. Two of the available approaches to encode the variables of the GA are described in the following sections.

#### 4.1.2 Binary encoded variables

There are many ways to encode and decode the variables of individuals (Engel 2010, p. 586). Usually a binary string encoding containing 1's and 0's is used. Binary encoding enables the selection, crossover and mutation operations of the GA to be

performed with relative ease.

As a simple example to illustrate binary encodings, consider the case where a mathematical problem needs to be optimized. Integer values between 8 and 16 are deemed to be solutions, or individuals in GA terms. Any integer can be represented by a binary string which is 4 characters long. A few encoded integers are shown below:

1.  $12 \rightarrow 1100$
2.  $14 \rightarrow 1110$
3.  $9 \rightarrow 1001$
4.  $13 \rightarrow 1101$

Binary encodings allow for the use of schemata during the optimization routine. Schemata is described by a set of chromosomes with certain common features (Chong et al. 2013, p. 292). More specifically, schema is a set of chromosomes which have 0's and 1's in certain locations, while the rest of the binary string is denoted by a \* symbol. Hence the binary string can now contain one of three symbols, 0, 1 or \*. In the case of a binary string which is 4 digits long, it can be noted that the schema  $1*01$  can refer to either 1101 or 1001.

Schemata can be used to identify good solutions. For example, when it is known that a certain type of cross-section produces good results, a schema can be used to constrain all elements in a structure to have that cross-section. This is useful as it eliminates bad solutions early in the optimization process.

One disadvantage of binary encoding as described by Chong et al. (2013, p. 297) is that it may cause the problem to be more complex than what it actually is. To illustrate this in mathematical terms, consider  $g(x)$  to represent the binary decoding function and  $x$  to be a chromosome. The objective function that is being optimized is not the same as the original,  $f(x)$ , but rather the combination of  $f$  and  $g$ . Now the optimization problem can be described as

$$\text{maximize} \quad f(g(x)) \text{ with respect to } x \quad (4.1)$$

The newly obtained optimization problem may be more complex than the original. For example, by having additional maximizers the search for the global maximum



may become more difficult.

When considering more complex optimization problems, it should be realised that encoding the problem also becomes more complex. For example, a structure with many elements and various configurations will have to be represented by a significantly long binary string. In addition, the longer the string becomes, the higher the probability that a crossover or mutated string has no meaning and should be discarded from the optimization. It is possible, however, to consider a method by which only part of the solution representation is binary encoded and the remainder is encoded with another type of encoding. For example, a discrete variable in a problem may be binary encoded, where a continuous variable may not be encoded at all. The case where a variable is not encoded is referred to as a real-value variable, discussed in the next section.

### 4.1.3 Real-value variables

The additional complexity related to a binary encoded variable provided motivation for the development of GAs to accommodate variables which do not use any sort of encoding. Instead they operate directly on the original optimization problem.

The approach for the real-value variable is the same as for the binary encoded one. The only real differences occur in the crossover and mutation operations. There are a few strategies which cater for the differences in these operations.

1. Crossover

The simplest option for a real-value crossover is to use averaging (Chong et al. 2013, p. 297). For a more complex problem a random combination of characteristics is also a feasible option, provided that the result can also be considered as a solution to the optimization problem.

2. Mutation

For a real-value variable, mutation is applied by randomly changing a value by a small percentage, provided that the result is still considered a feasible solution to the problem. Patton et al. (1994) suggested that number creep can be used for

mutation. Number creep entails that all values within a solution are “creeped” up or down by a small, random amount.

#### 4.1.4 Disadvantages of the genetic algorithm

As with any algorithm, the GA has a number of disadvantages which must be taken into consideration before using the algorithm. These disadvantages are listed below:

1. An important disadvantage is the care that must be taken when defining the the problem representation. This is related to the encoding of variable to represent the problem. For example, if binary encodings are to represent continuous variables, the length of the binary string can limit the precision of the variable (Fogel 2005, p. 147).
2. The choice of parameters for the GA can have a significant influence on the end result (Grefenstette 1994, p. 69). The parameters such as population size, mutation rate and maximum number of function evaluations are usually chosen by means of trial and error.
3. Another concern with the GA is that the algorithm may be subjected to premature convergence (Andre et al. 2001). This is related to the random generation of the initial population and the choice of the GA parameters. In the case of a small population size, the probability exist that a number of unacceptable solutions or one very good solution resides in the population. In each of these cases, the algorithm may struggle to find better solutions and maintain diversity during the optimization.

#### 4.1.5 Multi-objective adaptation

The GA as discussed thus far caters for the case where only one objective function is present in the problem. Allowance in the GA for use in multi-objective problems was first made by Srinivas et al. (1994) and improved by Deb, Agrawal, et al. (2000) to be known as the non-dominated sorting genetic algorithm (NSGA-II). The remainder of the section describes the NSGA-II as presented by Deb, Agrawal, et al. (2000).

The NSGA-II uses a specialized selection scheme to select a range of solutions within the population to act as parents for the next generation. When the initial parent

## 4.1 Genetic algorithms

---

population,  $P_0$ , is generated, the first step in this selection scheme is to rank each solution in the population according to the level of non-domination. This entails comparing all the solutions' objective values to one-another in a procedure called non-dominated ranking. This procedure identifies various fronts in the objective space where one of these fronts is the current pareto front.

The second step of the specialized selection scheme is to perform a population density evaluation. This involves gathering information regarding the spread of solutions surrounding a particular position in the so-called objective space. The average distance between the solutions adjacent to the point under consideration is used as a measure of the density. This measure is typically termed crowding distance.

Once a non-domination rank and crowding distance value is assigned to each solution in the population, the population is finally sorted based on these two parameters. This assists the selection process to achieve a uniformly spread-out pareto optimal front. Solutions with lower dominance ranks and members of prior fronts are preferred. Between solutions with the same rank, the one with the larger crowding distance is preferred. This sorting mechanism is termed the crowded comparison operator,  $\prec_n$ . From this population, the first child population,  $Q_0$ , is created by means of tournament selection, crossover and mutation.

The procedure is different for generations succeeding the first generation. First a combined population,  $R_t$ , is created, where  $t$  indicates the generation number and is greater than one. The next parent population,  $P_{t+1}$ , is formed by sorting  $R_t$  with respect to non-domination and adding the lowest ranked solutions until the desired population size is exceeded. Next all the solutions originating from the last considered front are sorted according to their crowding distance and added to the new parent population until it reaches the population size. Now the next child population,  $Q_{t+1}$ , can be created by means of tournament selection, crossover and mutation. It is important to note that the tournament selection uses the crowded comparison operator,  $\prec_n$ .

## 4.2 Simulated annealing

Simulated annealing (SA) is an optimization technique based on the gradual cooling of a solid which is heated above its melting point (Haupt et al. 2004, p. 187). SA has been used by a number of researchers for finding solutions to structural problems. For example Erbatur (2002) used an efficient SA algorithm for complex structural optimization problems, while Torbaghan et al. (2013) and Sonmez (2007) used SA for size and shape structural problems respectively.

When a solid material, for example metal, is heated to a temperature that exceeds the melting temperature of the material, the atoms in the molten material are free to move with respect to each other. When the heated material's temperature begins to decrease, the movement of these atoms become restricted and the atoms order themselves to finally be in a state of lowest energy. This cooling process is usually controlled to be as slow as possible in order to have the least amount of energy in the material, producing a result of better quality. This process of cooling at a slow rate is known as annealing (Rao 2009, p. 702).

In this discussion of the SA algorithm, background information regarding how the algorithm functions, its parameters and the process followed during an optimization is described. A brief discussion of the multi-objective adaptation made to the SA is also presented.

### 4.2.1 Background

Lamberti (2008) notes that SA follows a rather simple optimization strategy. A trial solution is randomly generated and the fitness function is evaluated at this point. In the case where the trial solution is deemed to be infeasible, the solution is rejected and a new trial solution is evaluated. If a solution is found that is a better solution than the current best solution, the current best solution is updated. In this manner, the best solution resulting from each stage of the optimization procedure is stored and used as a measure for the other trial solutions. If a trial point is considered to be feasible but not a better solution than the current best solution, the point is either accepted or rejected by the algorithm based on a probabilistic criterion which

## 4.2 Simulated annealing

---

estimates whether this point will lead to a better solution in the coming trial solutions.

This probability criterion is determined by a parameter known as the “temperature” parameter. This parameter may be an estimated target solution (Lamberti 2008) or a combination of randomly generated solutions (Rao 2009). At the beginning of the algorithm, a large temperature parameter is selected and it is reduced based on a so-called “cooling schedule”. The acceptance probability gradually reduces to zero as the temperature is reduced (Lamberti 2008). Torbaghan et al. (2013) note that the acceptance of a solution by using the temperature parameter allows for “uphill” climbing which potentially saves the SA algorithm from becoming stuck at a local optimum.

A number of variations of the SA has been developed and tested by researchers for different applications. During these developments, it became clear that the effectiveness of any SA is dependent on three factors namely:

1. Choice of the temperature
2. Algorithm design
3. Extent of the problem that needs to be solved

Rao (2009, p. 705) presents five features of the SA method. These may be used as considerations when determining whether or not an SA should be used for a certain optimization problem. These features can be listed as:

1. The quality of the final solution is not affected by the initial trial solutions, but the computational effort may increase with poor starting solutions.
2. Due to the discrete nature of SA, the convergence characteristics are not affected by the continuity or differentiability of functions.
3. The design variables do not need to be positive.
4. SA is applicable to mixed-integer, discrete or continuous problems.
5. It utilizes objective functions in addition to the normal upper and lower bound conditions.

### 4.2.2 Procedure of simulated annealing

A general SA algorithm works on the simple procedure of starting with an initial solution as well as the preselected values for the temperature and maximum number of iterations. The procedure described here is adapted from Rao (2009, p. 703).

The temperature parameter is controlled by Boltzmann's probability distribution which implies that the energy,  $E$ , of a system in thermal equilibrium is probabilistically distributed according to equation 4.2.

$$P(E) = e^{-E/kT} \quad (4.2)$$

Where  $P(E)$  denotes the probability of achieving an energy level of  $E$ ,  $k$  denotes Boltzmann's constant and  $T$  denotes the temperature parameter. It can be deduced from equation 4.2 that at high temperatures the system has a high probability to be at any energy state, but at lower temperatures this probability decreases. The Boltzmann's probability distribution is used for function minimization in the same way as for thermodynamic systems.

To illustrate this procedure, let the current solution, for example a truss or a frame structure, be denoted by  $X_i$  with the corresponding value of the objective function, for example the weight of the structure, denoted by  $f_i = f(X_i)$ . The energy in the system at the current state,  $E_i$ , can be determined by equation 4.3.

$$E_i = f_i = f(X_i) \quad (4.3)$$

According to the Metropolis criterion (Metropolis et al. 1953), the probability of the next solution,  $X_{i+1}$ , depends on the difference in the energy state. This can be expressed in terms of the objective function values given by equation 4.4.

$$\Delta E = E_{i+1} - E_i = \Delta f = f_{i+1} - f_i = f(X_{i+1}) - f(X_i) \quad (4.4)$$

The new solution,  $X_{i+1}$ , can be found by using the Boltzmann's probability distribution. This is shown in equation 4.5.

$$P[E_{i+1}] = \min \left\{ 1, e^{-\Delta E/kT} \right\} \quad (4.5)$$

## 4.2 Simulated annealing

---

It can be seen from the above equation that Boltzmann's constant serves as a scaling factor, therefore, for a SA implementation it can be chosen as 1. By doing so the symbol,  $k$ , can effectively be removed from equation 4.5.

A characteristic of equation 4.5 is when  $\Delta E < 0$ , the result is 1. This occurs when  $f_i$  is greater than  $f_{i+1}$ . In the context of function minimization this means that  $X_{i+1}$  is a better solution and will be accepted.

For the case when  $\Delta E > 0$ , the new solution is classified as a worse solution. In typical optimization procedures this solution would just be discarded, but by applying equation 4.5 the result is not always the same. The probability depends on the values of  $\Delta E$  and  $T$ . If  $T$  is large, the probability will be high for larger values of  $\Delta E$ . This means that at high temperatures, even remarkably worse solutions are likely to be accepted. At low temperatures, this probability will decrease significantly.

The procedure of this algorithm can be graphically presented as shown in figure 4.3. The values of  $n$  and  $c$  denote the maximum amount of iterations and the percentage by which  $T$  is reduced in-between iterations respectively.

## 4.2 Simulated annealing

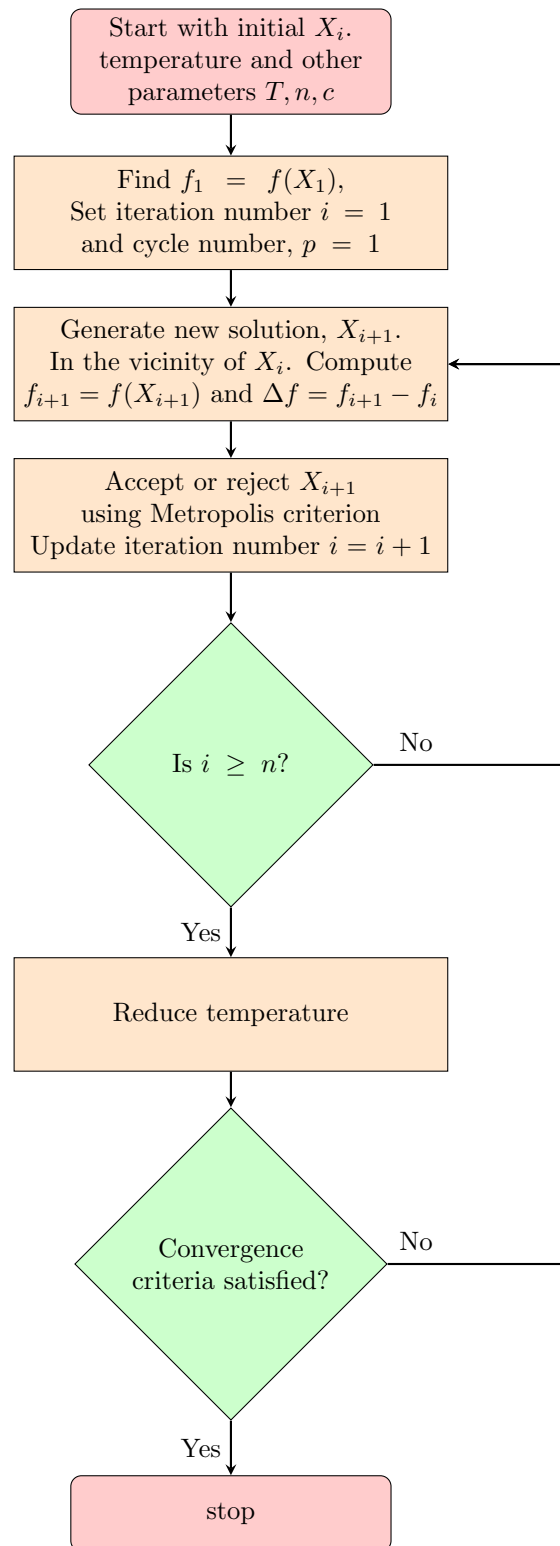


Figure 4.3: The SA process (Rao 2009, p. 706)



### 4.2.3 Disadvantages of simulated annealing

When considering the SA algorithm for an optimization routine, the following disadvantages must be taken into consideration.

1. The main disadvantage of the SA algorithm is that it is known to be a slow algorithm (Rutenbar 1989). This is due to the algorithm's nature to consider many configurations in order to reach a good solution..
2. The nature of SA allows it to only be applicable to specific kinds of problems (Rutenbar 1989). For example, problems which combine different variable types can not be solved by means of a SA algorithm.
3. When multi-objective problems are considered, the SA algorithm typically use a population of solutions. The use of a population in SA may lead to redundant searches which degrades its performance (Nam et al. 2000).

### 4.2.4 Multi-objective adaptation

There exist several modifications made by various researchers to the standard SA algorithm to accommodate multi-objective problems. The majority involve a single change in the way the solutions are compared to one another.

For example, Bandyopadhyay et al. (2008) suggested a non-dominance comparison similar to the method used by the NSGA-II. Solutions are then replaced based on whether or not they are non-dominated as opposed to comparing the single objective value.

## 4.3 Particle swarm optimization

Particle swarm optimization (PSO) was first presented by Eberhart et al. (1995) to mimic the natural behaviour of swarms, flocks or schools of animals in an optimization routine. It was found by Kennedy (2011) that the PSO algorithm performs well on the same test functions as that of a GA, justifying why it is considered in this study.

This section provides background information regarding the PSO algorithm which includes the associated terminology and parameters. The procedure followed during

## 4.3 Particle swarm optimization

an optimization and its multi-objective adaptation is discussed in sections 4.3.2 and 4.3.4 respectively.

### 4.3.1 Background

The term particle refers to an individual within a swarm, for example, a bird in a flock. Each particle behaves in an individual way using its own knowledge and also the collective knowledge of the swarm. If one particle discovers food, or a good solution, the rest of the swarm is guided by this information and start to follow the general direction towards the good solution, irrespective of how far away it may be (Couceiro et al. 2015, p. 2). Optimization methods that are based on swarm intelligence are termed behavioural inspired algorithms.

The size of the swarm is generally preselected and particles are generated at random positions in the so-called “solution space”. Each particle is awarded two characteristics, namely, a position and a velocity (Kiranyaz et al. 2014, p. 46). During the optimization, each particle remembers the best position corresponding to the best solution it has discovered. Particles also communicate with each other to share information regarding good positions and they adjust their respective positions and velocities accordingly. The behaviour of particles is based on the following factors (Rao 2009, p. 709):

1. Cohesion - Stick together.
2. Separation - Do not move too close to each other.
3. Alignment - Follow the general heading of the swarm.

### 4.3.2 Procedure of a particle swarm optimization

The procedure described here is adapted from Rao (2009, p. 710). It illustrates how the PSO algorithm manages to locate a good solution, for example a light structure, from start to finish. Consider a maximization problem expressed as:

$$\begin{array}{ll} \text{Maximize} & f(X) \\ \text{With} & X^l \leq X \leq X^u \end{array} \quad (4.6)$$

### 4.3 Particle swarm optimization

Where  $X^l$  and  $X^u$  denote the lower and upper boundaries of  $X$  respectively. These boundaries can be seen as the range a node within a structure can be move during a shape optimization problem or the amount of available cross-sections for a size optimization problem. By keeping the format in Rao (2009), the procedure can be described by the following steps.

1. Start by selecting the size of the swarm, denoted by  $N$ . To reduce the total number of function evaluations, the value of  $N$  may be reduced, although a too small value of  $N$  will cause the algorithm to consume more time to arrive at a solution. Usually the value of  $N$  is chosen to be between 20 and 30.
2. The initial population of solutions,  $X$ , is generated within prescribed boundaries. The size of the population is equal to  $N$ . Every particle is assigned a different initial solution. In other words particle  $j$  is assigned  $X_j(0)$ , where 0 indicates the initial solution. Now the objective function value of the solution attributed to each particle is determined. With respect to structural optimization, during this step each particle is assigned a candidate structure and its weight is calculated.
3. Velocities are now assigned to each particle. At the beginning of the algorithm, all velocities are assumed to be zero. The iteration number,  $i$ , is set as 1.
4. In the  $i^{\text{th}}$  iteration, the following steps are executed by particle,  $j$ :
  - (a) Two parameters are obtained by the particle. The first is the best solution that the particle has come across during its search, this particular solution,  $X$ , is named  $P_{best}$ . The second is the best solution that has been found by any particle within the swarm, this is termed to be  $G_{best}$ . The fitness values of both  $P_{best}$  and  $G_{best}$  are obtained and denoted as  $f(P_{best})$  and  $f(G_{best})$  respectively.
  - (b) Now the velocity of the particle is determined using equation 4.7.

$$V_j(i) = V_j(i-1) + c_1 r_1 [P_{best} - X_j(i-1)] + c_2 r_2 [G_{best} - X_j(i-1)] \quad (4.7)$$

Where  $c_1$  and  $c_2$  are individual and social learning rates respectively while  $r_1$  and  $r_2$  are uniformly distributed random numbers in the range of 0 to 1. The parameters  $c_1$  and  $c_2$  represent the relative importance of the position of the particle to the position of the swarm. The values of  $c_1$  and  $c_2$  are typically selected to be 2. This selection ensures that the particles

### 4.3 Particle swarm optimization

---

have a 50% chance to overshoot the target solution. In a structural size optimization context, the velocities can be considered a weighted combination of the cross-sections of the current candidate structure and the best structure present in the entire swarm as well as the current particle.

- (c) Finally, the new solution can be found of the  $j^{\text{th}}$  particle for the  $i^{\text{th}}$  iteration by using equation 4.8.

$$X_j(i) = X_j(i - 1) + V_j(i) \quad (4.8)$$

The value of the objective function is determined for each new solution,  $X_j$ .

5. The current solution is then checked for convergence. When the positions of all the particles converge to the same set of values, the method is assumed to have converged. If convergence is not achieved, step 4 is repeated for the next iteration  $i = i + 1$ .

The above described procedure for the PSO algorithm can be graphically expressed as shown in figure 4.4.

### 4.3 Particle swarm optimization

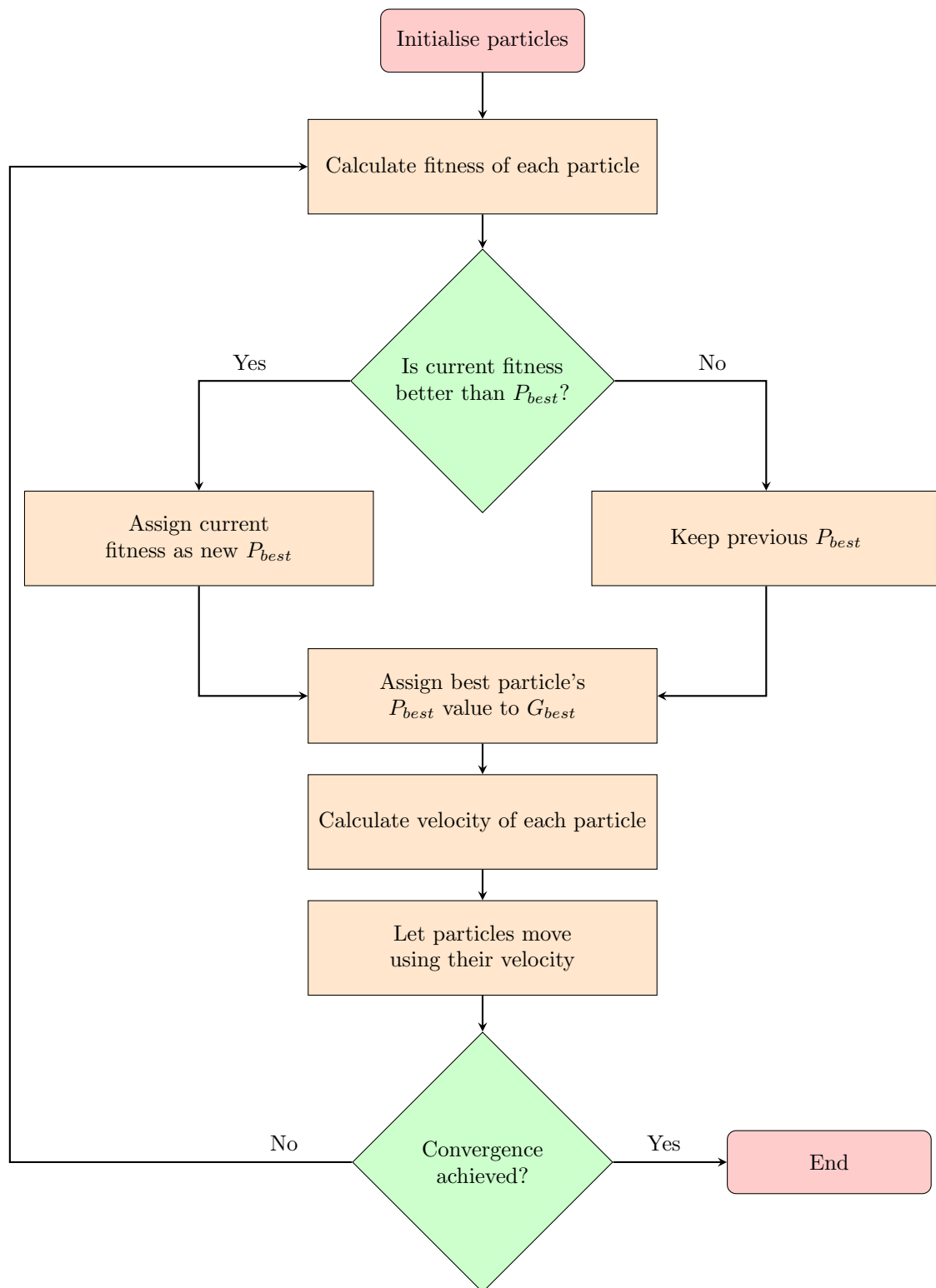


Figure 4.4: The PSO algorithm procedure (McCulloch 2016)

### 4.3.3 Disadvantages of particle swarm optimization

The PSO algorithm has the following disadvantages which must be considered before implementing it to solve optimization problems:

1. PSO is known to have a tendency for premature convergence on local optimum points in large search spaces (M. Li et al. 2014). By converging prematurely, the algorithm stops while the optimum result still needs to be located. This leads to the possibility of the PSO yielding sub-optimal results.
2. Bratton et al. (2007) note that there is no fixed swarm size which suits all problem definitions. Therefore, the PSO algorithm must be calibrated by trial and error to determine a suitable swarm size for each problem.
3. The PSO algorithm may also have a slow convergence rate in small search areas (Ab Wahab et al. 2015). This weak local search ability causes the PSO to struggle in the scenario of refining a good solution in order to reach the optimum solution.

### 4.3.4 Multi-objective adaptation

As in the previous algorithms, allowance has also been made to apply the PSO algorithm to a multi-objective problem. Although there are a number of variations available, only the adaptation made by Sierra et al. (2005) to develop the multi-objective particle swarm optimizer (MOPSO) is discussed here.

In the MOPSO introduced by Sierra et al. (2005), the best particles,  $P_{best}$  and  $G_{best}$ , are replaced by a number of non-dominated best particles. The size of the allowable particles in this set of solutions is fixed to avoid excessive results. The number of best particles is prescribed to be smaller than or equal to the swarm size. This allows the result of the algorithm to yield a pareto set of solutions. In addition, a crowding factor is also introduced for the comparison of particles to ensure that a spread of solutions is obtained within the objective space. This approach is analogous with that of the NSGA-II.

Furthermore, the concept of  $\epsilon$  - dominance is also used. This concept allows similar solutions to be filtered out of a population and promotes the search for a more diverse

#### 4.4 Ant colony optimization

The ACO algorithm has been applied to a number of structural optimization problems. For example, Kaveh and Talatahari (2009a) used it in combination of other algorithms to optimize truss structure and Camp and Bichon (2004) optimized the design of space trusses.

46

### **4.4.1 Background**

ACO is, as the name suggests, based on the cooperative behaviour of an ant colony to locate food or other resource. The process is fairly simple, thousands of ants leave the nest in search of food, each following a unique path. As each ant moves, it releases a pheromone. This can be interpreted as a trail that other ants can follow. If an ant is successful in locating food, the amount of pheromone on its path is increased. Ants follow the paths with the most pheromone as they are aware that such paths lead to food. The pheromone on paths that are not frequently taken fades. This process is repeated until a satisfactory resource is found by the ant colony.

For the formulation of ACO, the behaviour of the ants and the fading of pheromone must be explicitly defined. The following discussion is adapted from Rao (2009, p. 715).

By considering figure 4.6, it is clear that the ACO process is approached as a multi-layered problem. The number of layers correspond to the number of design variables and the number of nodes that each layer contains is the amount of discrete values that the layer can assume.



## 4.4 Ant colony optimization

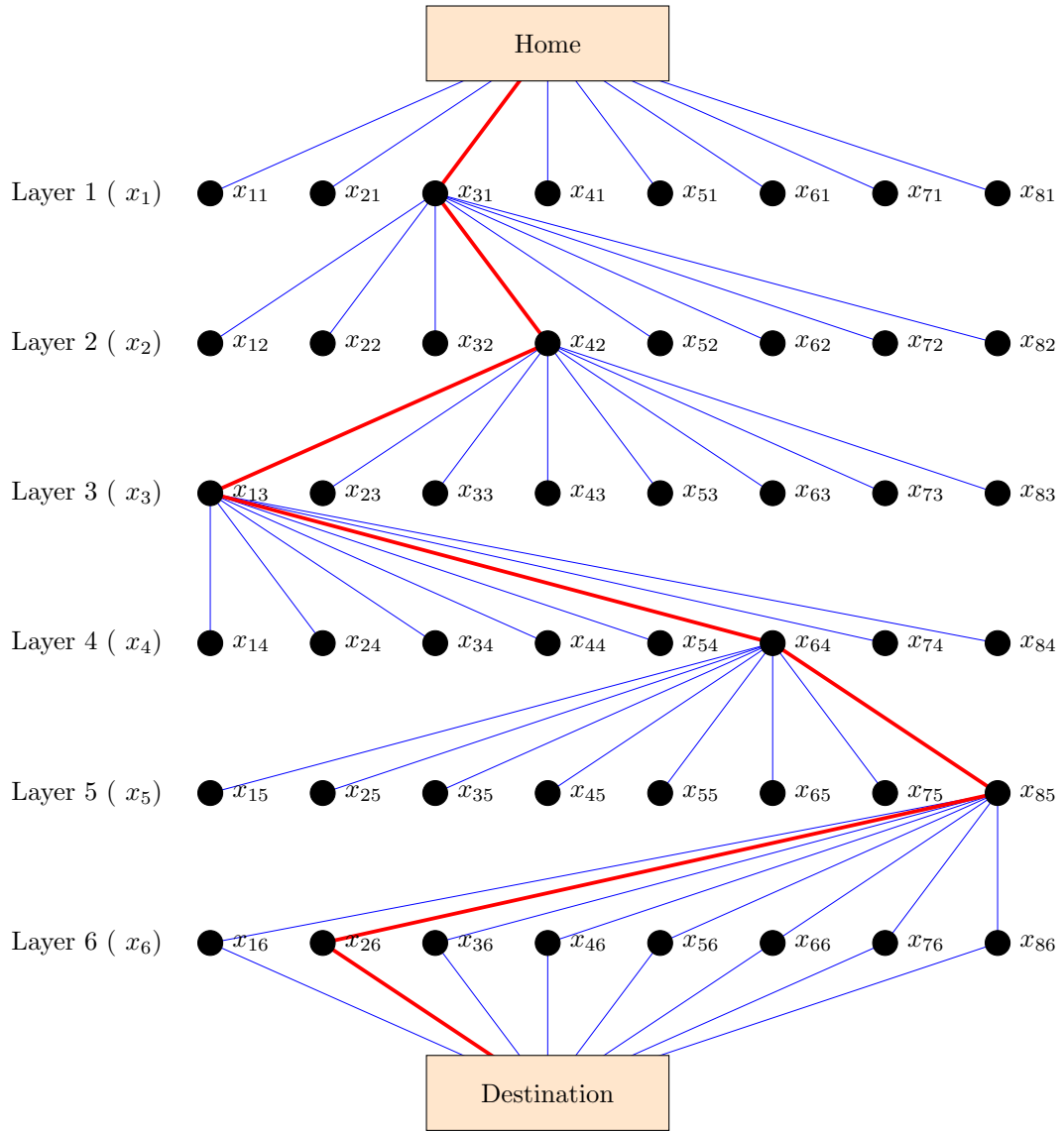


Figure 4.6: The ACO process (Rao 2009, p. 714)

The colony consists of  $N$  ants which all start from the home node and travel through the layers until the final layer to end at the so-called destination node. This process is repeated for every iteration, while the pheromone from the previous iteration is still present on the paths. Each ant can select only one node in each layer based on probability, which is calculated using equation 4.9.

$$p_{ij}^{(k)} = \begin{cases} \frac{\tau_{ij}^\alpha}{\sum_{i \in N_i^{(k)}} \tau_{ij}^\alpha}, & \text{if } j \in N_i^{(k)} \\ 0 & \text{if } j \notin N_i^{(k)} \end{cases} \quad (4.9)$$

Where an ant,  $k$ , situated at node  $i$  uses the pheromone trail,  $\tau_{ij}$ , to calculate the probability of choosing  $j$  as its next node. The importance degree of pheromones is denoted by  $\alpha$  and the set of neighbourhood nodes adjacent to ant  $k$  when its situated at node  $i$  is denoted by  $N_i^{(k)}$ . This neighbourhood of nodes contains all the nodes directly connected to node  $i$ , except the last visited node in order to avoid the ant returning to the previous node.

The nodes visited by an ant along the path represent a candidate solution. In the context of structural optimization, each node can be considered as a variable of the problem such as nodal positions and member sizes. When the path is complete, the ant releases pheromone on that path on it's way back to the nest. The amount of pheromone,  $\Delta\tau_{ij}$ , released on each section of its path is calculated based whether or not the section forms part of the best path during that iteration. In the case the section forms part of the best path,  $\Delta\tau_{ij}$  is calculated using the ratio of the best and worst objective function values of that iteration, otherwise no pheromone is deposited. This pheromone updating procedure is mathematically expressed in the upcoming section.

Another aspect of ACO is that there is a scheme which represents the fading of pheromone. By doing so it favours other paths with higher pheromone to be explored by other ants. As an ant moves to its next node, the amount of pheromone released by the ant on its path is reduced. This scheme will become clearer during the procedural discussion of the algorithm in section 4.4.2.

At the start of the optimization process, all possible paths are given the same amount of pheromone. This leads to the ants randomly selecting a path to follow in the beginning, until certain paths contain more pheromone. The optimization procedure is terminated when a maximum number of iterations have been reached or the best solution could not be improved for a selected number of continuous iterations. When

all the ants have completed their paths, the path with the most pheromone is deemed to be the best solution.

#### 4.4.2 Procedure of algorithm

The procedure of a typical ACO algorithm can be described by the following steps:

1. Start with a suitable size for the colony,  $N$ . Generate a set of discrete values for each of the  $n$  design variables. Name these values as in figure 4.6, with each value denoted by  $x_{ij}$ , where  $i$  and  $j$  denote the index of the design variable and the index of the value of that variable in the generated set respectively.  $i = 1, 2, \dots, n$  and  $j = 1, 2, \dots, p$  with  $p$  being the number of values in the discrete set. Assume an equal amount of pheromone  $\tau_{ij}^1$ , where the superscript indicates the iteration number, along all paths. Set the iteration number,  $\alpha = 1$ .

2. (a) Compute the probability of selecting a trail as:

$$p_{ij} = \frac{\tau_{ij}^\alpha}{\sum_{m=1}^p \tau_{im}^\alpha}; i = 1, 2, \dots, n; j = 1, 2, \dots, p \quad (4.10)$$

- (b) The path chosen by an ant is determined by means of a randomly generated number between 0 and 1. To use this number, the cumulative probability ranges associated with different paths are calculated.
3. (a) Generate  $N$  random numbers between 0 and 1, one for each ant,  $r_k$ . The path that ant  $k$  is to assume is the one whose probability range, determined in the previous step, includes the value of  $r_k$ .
  - (b) Repeat the step 3a for all design variables  $i = 1, 2, \dots, n$ .
  - (c) Evaluate the objective function for paths chosen by all the ants. Each complete path corresponds to the solution,  $X^k$ . Hence, compute  $f_k = f(X_k)$  with  $k = 1, 2, \dots, N$ .
  - (d) Determine the best and worst paths among the calculated fitnesses, this yields  $f_{best}$  and  $f_{worst}$ .
4. Test if convergence was achieved by checking whether or not all the ants take the same path. In other words, if all the ants found the same solution. In the case where convergence was not achieved, assume all the ants return to the

## 4.4 Ant colony optimization

nest to start the next iteration. Increment the iteration number and update the pheromone on different trails. These are shown in equations 4.11 to 4.14

$$\alpha = \alpha + 1 \quad (4.11)$$

$$\tau_{ij}^{(\alpha)} = \tau_{ij}^{(old)} + \sum_k \Delta\tau_{ij}^{(k)} \quad (4.12)$$

Where

$$\tau_{ij}^{(old)} = (1 - \rho)\tau_{ij}^{(\alpha-1)} \quad (4.13)$$

$$\Delta\tau_{ij}^{(k)} = \frac{\zeta f_{best}}{f_{worst}} \quad (4.14)$$

$\tau_{ij}^{(old)}$  represents the pheromone left from the previous iteration, after evaporation has occurred.  $\Delta\tau_{ij}^{(k)}$  denotes the pheromone deposited by the best ant,  $k$ , on its path and the summation extends over all the ants which chose the globally best path, if there are more than one. The pheromone evaporation rate factor,  $\rho$ , is typically in the range of 0.5 to 0.8 and the amount of pheromone deposited is determined with equation 4.14, where  $\zeta$  is a parameter that is used to control the scale of the pheromone amount. The larger  $\zeta$ , the more pheromone is deposited. This allows for more pheromone to be added to the paths that yield better solutions.

Now, with all the updated values the ants can be released for their next search. This implies that the procedure is repeated from step 2 onward. The procedure is carried out until convergence or a maximum number of iterations is reached.

### 4.4.3 Disadvantages of ant colony optimization

As with the previously discussed algorithms, the ACO algorithm also has a few disadvantages. These are described below:

1. The choice of parameters of the ACO algorithm needs to be established through iteration (Abbaspour et al. 2001). The more iterations used, the better suited the parameters for the problem at hand.
2. The sequences of random decisions taken by the ants during each iteration is not independent (Selvi et al. 2010). This may cause the algorithm have an inherent bias during the optimization.

3. The time to convergence is uncertain (Selvi et al. 2010). If a maximum number of iterations is not prescribed, the optimization may continue for a significant amount of time before all the ants arrive at the same solution. Convergence is however guaranteed irrespective of the time required to achieve it.

#### 4.4.4 Multi-objective adaptation

The standard ACO algorithm has been adapted for multi-objective problems in a number of different ways. To illustrate how this can be done, the method devised by Thantulage (2009) and summarised by Ariyasingha et al. (2015) is discussed.

Thantulage (2009)'s adaptation is called pareto strength ant colony optimization (PSACO). The PSACO is based on the normal ACO with the difference being that the pheromone trail is updated using the non-dominance concept, similar to the NGA-II discussed in section 4.1.5.

The pheromone updating procedure is complemented by including two sets of solutions to the algorithm. Namely, a population,  $P_t$ , and an archive,  $A_t$ . Solutions produced by the current iteration,  $t$ , are kept in  $P_t$  and  $A_t$  contains the globally best non-dominated solutions.

With these newly added parameters, the pheromone update step can be performed as in the standard ACO algorithm by replacing the single fitness value with a value which represents the quality of a solution,  $Q_i$ . This value is determined by combining the fitness values of a solution with a density value. This density value is calculated using a method which determines how close solutions are to one another known as the k-th nearest neighbour method.

At the end of each iteration, a new archive is created by gathering all the non-dominated solutions from the current population,  $P_t$ , and the current archive,  $A_t$ . If the size of the new archive exceeds the prescribed amount, it is truncated by removing several of the worse performing solutions until the archive is of the prescribed size.

## 5. Software implementation

The theoretical basis with regard to basic optimization, chapter 2, structural optimization, chapter 3, and algorithms to solve optimization problems, chapter 4, have been covered. This chapter discusses how these concepts are combined to develop a software package that can solve both single- and multi-objective problems for both two- and three-dimensional truss and frame structures.

The first section discusses the development of a finite element analysis (FEA) module required to determine whether or not a structure meets the constraints of the optimization problem. The second section discusses the development of an optimization module with which a given optimization problem can be solved using a chosen algorithm. Both these modules must work together to successfully optimize a structure. Lastly, the development of a visualization module, which simplifies the model creation process and prevents input errors, is discussed.

### 5.1 Finite element analysis module

In this section the development of a finite element analysis (FEA) module is discussed. The FEA module is required to calculate the nodal displacements and element forces that arise in a structure under certain load conditions. These displacements and forces are used to determine whether or not a structure violates the prescribed constraints of the optimization problem. This information is then used to determine if a generated structure is a suitable solution to a given optimization problem.

For the purposes of this study, only linear elastic analyses are considered. This is done taking the iterative nature of both optimization algorithms and higher-order analysis methods into consideration. If a higher-order analysis method is used, the optimization procedure would be subjected to a significant amount of additional computation. However, the use of a second-order or non-linear analysis would improve analysis results.

It is important to note that, given the iterative nature of an optimization algorithm and the amount of structures which are present in the population of

## 5.1 Finite element analysis module

each iteration, this FEA module must be able to manage its memory effectively to avoid an overconsumption. The Java programming language, which is used for this implementation, provides automatic clearing of memory which is no longer in use. With this knowledge, emphasis must be placed on ensuring that memory which is no longer required is made available for clearing.

The Java language utilises an object-oriented approach. This approach allows for easy allocation of various functionalities and attributes to certain objects. Furthermore, it allows for collectively creating a module where the definition and interaction of objects can be easily outlined and understood. For a typical FEA module, the main components required are shown in figure 5.1:

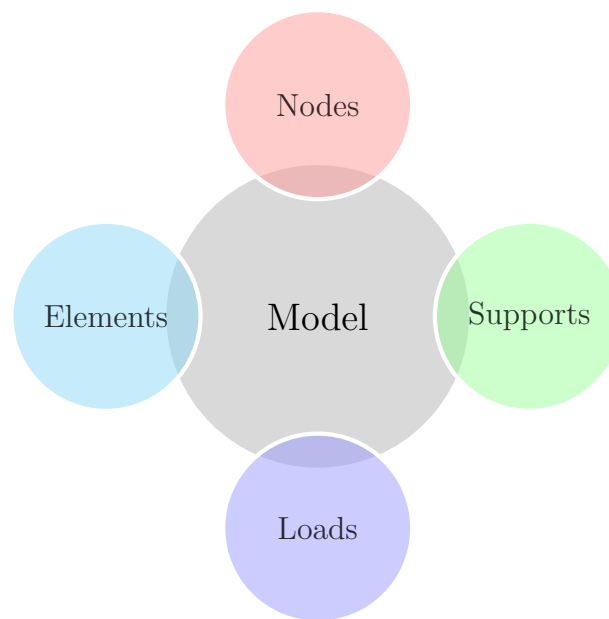


Figure 5.1: Main FEM components

The nodes of the model represent the nodal positions of points where elements are connected. This may be either two- or three-dimensional. The vertical axis is chosen as the y-axis as this complies with many element formulations found in literature.

Each element included in the model must consist of a material. The material provides the mechanical properties to the element that are used to determine its structural stiffness. In this study, all the elements in a structure will share the same material

## 5.1 Finite element analysis module

as typical in truss or frame structures.

The model also has various boundary conditions assigned to it. These are points where certain displacements are prescribed. For truss structures it relates to x, y and z displacements only and for frames the rotation about each axis is also included.

The remainder of the components require a more detailed discussion. These components are discussed in the following subsections.

### 5.1.1 Cross-sections

Every truss or frame element within the model must have a defined cross-section. A cross-section has a predefined shape and provides properties which relate to the behaviour of each element. A few well-known cross-sections are shown in figure 5.2.

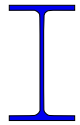
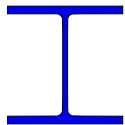
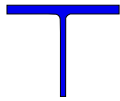


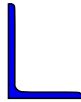



		
I-Section	H-Section	T-Section
		
Channel	Equal-leg angle	Unequal-leg angle
		
Circular hollow section	Square hollow section	Rectangular hollow section

Figure 5.2: Different cross-section forms

For the implementation of a cross-section, the class `CrossSection` is used to provide the required functionality. Properties such as area and moments of inertia are stored in a Map structure with predefined key values. The Map structure simplifies the process of retrieving values from a cross-section in the sense that all the key values are already defined and can easily be selected.



## 5.1 Finite element analysis module

This study only considers steel cross-sections. A number of steel cross-sections with standard dimensions and properties are typically available in a tabulated format. These tables are accommodated by using a database extension to the standard **CrossSection** class. The popular SQLite database library is utilised to provide this functionality (Hwaci - Applied Software Research 2017).

Since there are different forms of cross-sections, a class named **Profile** is used to differentiate between these different sections. The **Profile** class provides basic database functions including reading, writing and attributing the correct profile from the corresponding table in the database.

All the different section forms in the database have their own class which extends the **Profile** class and defines their specific properties, such as perimeter, leg length or height. For the purposes of this study, only I-, H- and angle-sections are included, but extension to other cross-section forms is possible. To illustrate the relation between these classes, a shortened unified modelling language (UML) diagram is presented in figure 5.3.

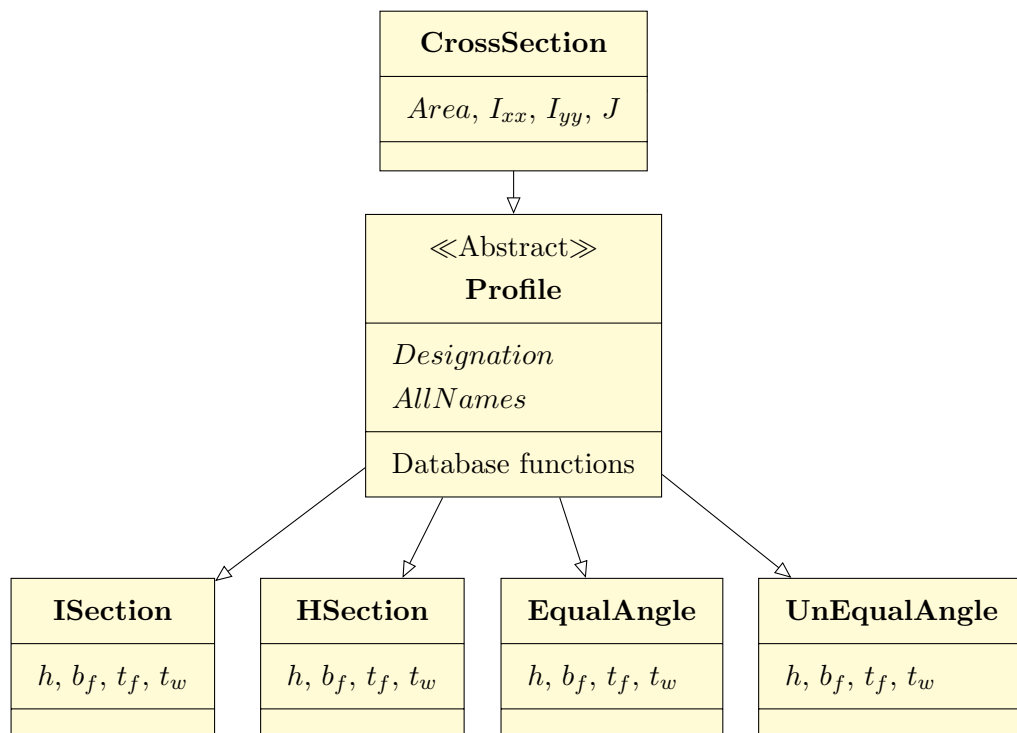


Figure 5.3: UML diagram illustrating cross-section class relation

### 5.1.2 Elements

In this study, only truss and frame structures are considered for the optimization, therefore, only these two elements require implementation. These element implementations are done keeping in mind the potential need for expandability to include additional elements in future studies.

To provide expandability, a superclass called **AbstractElement** is created to prescribe the basic requirements for all elements. This includes a name, material, cross-section and nodes. From this basis, the specific elements are added as subclasses.

Both two- and three-dimensional versions of the truss and frame elements are considered to cover a wide base of problems. The details of how this implementation is done for each element are discussed in the following subsections.

#### 5.1.2.1 Truss element

The truss element is a simple element which only makes allowance for axial forces. This leads to catering for one degree of freedom at each node to represent the axial force within the element.

By means of rotation, the one-dimensional element can be transformed to both two- and three-dimensional versions. By applying the rotation, the one degree of freedom is divided into two or three components, depending on the dimension of the element under consideration. Figures 5.4 and 5.5 illustrate how the two- and three-dimensional elements are represented.

## 5.1 Finite element analysis module

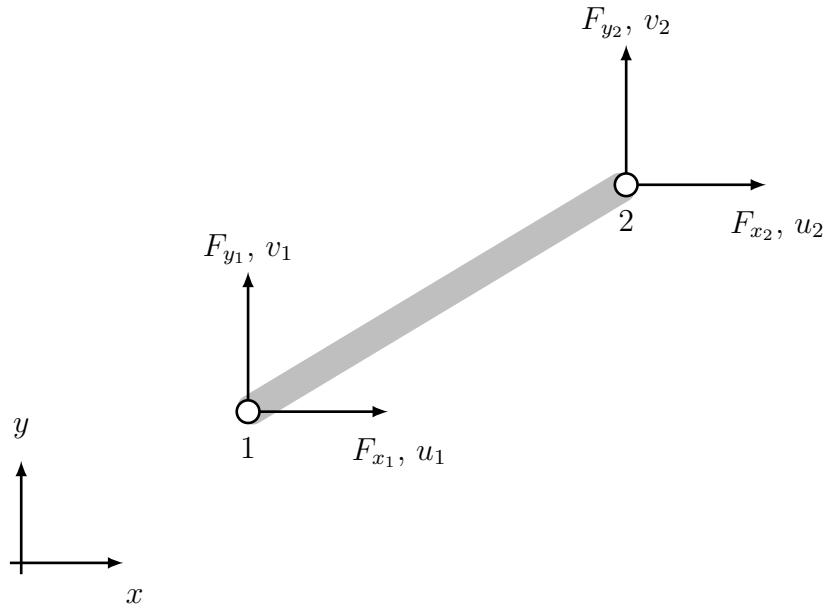


Figure 5.4: A two-dimensional truss element

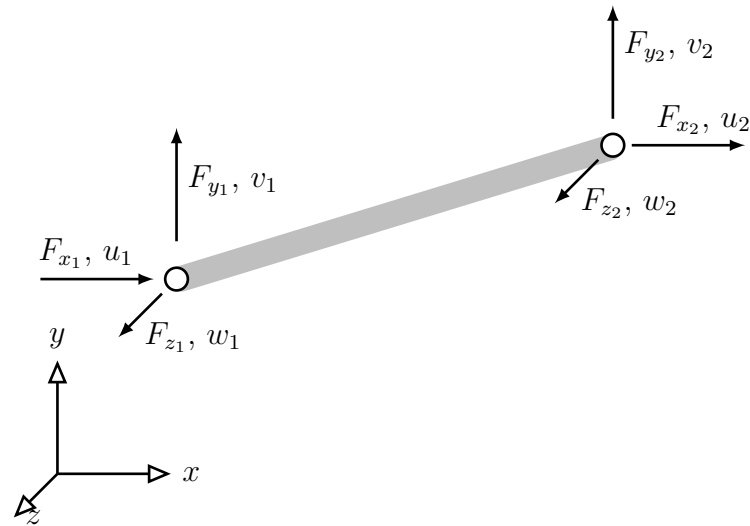


Figure 5.5: A three-dimensional truss element

The element matrices for both the two- and three-dimensional truss elements are shown in equations 5.1 and 5.2. The two- and three-dimensional rotation matrices are shown in equations 5.3 and 5.4 respectively (Logan 2011, pp. 87, 103).

## 5.1 Finite element analysis module

$$[k_{2Dtruss}] = \frac{A \cdot E}{L} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.1)$$

$$[k_{3Dtruss}] = \frac{A \cdot E}{L} \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.2)$$

$$[T_{2Dtruss}] = \begin{bmatrix} \cos \theta & \sin \theta & 0 & 0 \\ -\sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & \cos \theta & \sin \theta \\ 0 & 0 & -\sin \theta & \cos \theta \end{bmatrix} \quad (5.3)$$

$$[T_{3Dtruss}] = \begin{bmatrix} C_x & C_y & C_z & 0 & 0 & 0 \\ C_x & C_y & C_z & 0 & 0 & 0 \\ C_x & C_y & C_z & 0 & 0 & 0 \\ 0 & 0 & 0 & C_x & C_y & C_z \\ 0 & 0 & 0 & C_x & C_y & C_z \\ 0 & 0 & 0 & C_x & C_y & C_z \end{bmatrix} \quad (5.4)$$

With

$\theta$  as the angle of rotation in the 2D space

$$C_x = \frac{x_2 - x_1}{L}$$

$$C_y = \frac{y_2 - y_1}{L}$$

$$C_z = \frac{z_2 - z_1}{L}$$

$L$  being the length of the element

### 5.1.2.2 Frame element

For the analysis of frame structures, the implementation of the frame element is required. The frame element is an extension to the truss element in the sense that allowance is made for bending moments in addition to normal forces.

## 5.1 Finite element analysis module

There is a significant difference between the two- and three-dimensional frame element regarding the degrees of freedom. While the two-dimensional case consists of three degrees of freedom at each node, the three-dimensional case has six to include the torsional effect on the element. The element definitions for both cases are shown in figures 5.6 and 5.7.

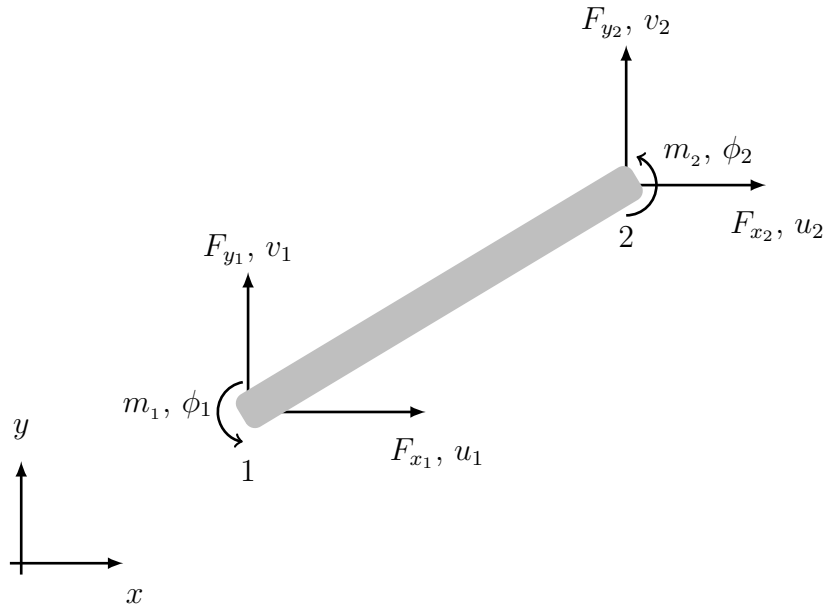


Figure 5.6: A two-dimensional frame element

## 5.1 Finite element analysis module

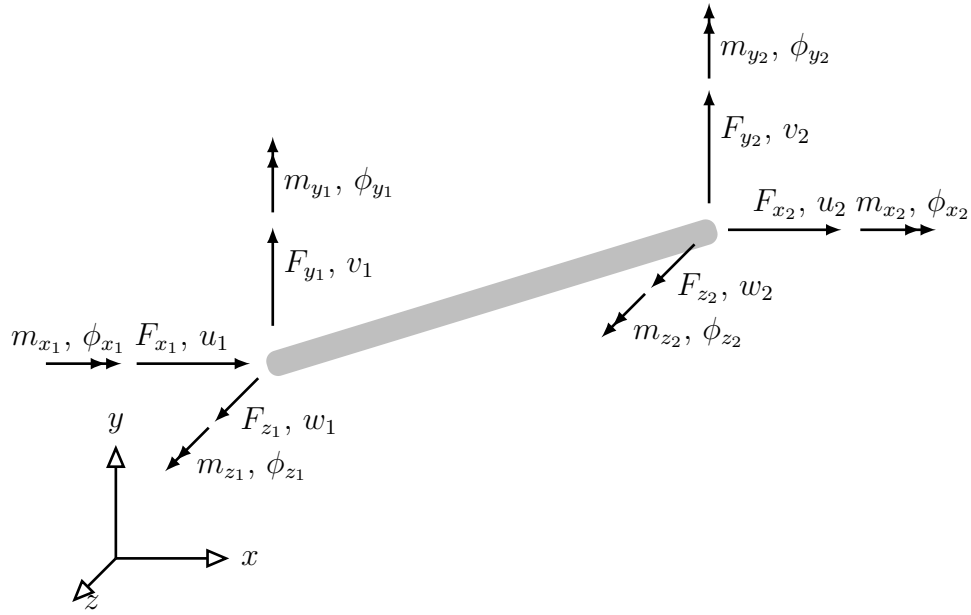


Figure 5.7: A three-dimensional frame element

The element stiffness matrices for the two- and three-dimensional cases are shown in equations 5.5 and 5.6 respectively (Logan 2011, pp. 238, 278).

$$[k_{2D}] = \begin{bmatrix} C_1 & \cdot & \cdot & -C_1 & \cdot & \cdot \\ \cdot & 12C_2 & 6C_2L & \cdot & -12C_2 & 6C_2L \\ \cdot & 6C_2L & 4C_2L^2 & \cdot & -6C_2L & 2C_2L^2 \\ -C_1 & \cdot & \cdot & C_1 & \cdot & \cdot \\ \cdot & -12C_2 & -6C_2L & \cdot & 12C_2 & -6C_2L \\ \cdot & 6C_2L & 2C_2L^2 & \cdot & -6C_2L & 4C_2L^2 \end{bmatrix} \quad (5.5)$$

$$[k_{3D}] = \begin{bmatrix} C_1 & \cdot & \cdot & \cdot & \cdot & \cdot & -C_1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 12C_2 & \cdot & \cdot & \cdot & 6C_2L & \cdot & -12C_2 & \cdot & \cdot & \cdot & 6C_2L \\ \cdot & \cdot & 12C_3 & \cdot & -6C_3L & \cdot & \cdot & \cdot & -12C_3 & \cdot & -6C_3L & \cdot \\ \cdot & \cdot & \cdot & C_4 & \cdot & \cdot & \cdot & \cdot & \cdot & -C_4 & \cdot & \cdot \\ \cdot & \cdot & -6C_3L & \cdot & 4C_3L^2 & \cdot & \cdot & \cdot & 6C_3L & \cdot & 2C_3L^2 & \cdot \\ \cdot & 6C_2L & \cdot & \cdot & \cdot & 4C_2L^2 & \cdot & -6C_2L & \cdot & \cdot & \cdot & 2C_2L^2 \\ -C_1 & \cdot & \cdot & \cdot & \cdot & \cdot & C_1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & -12C_2 & \cdot & \cdot & \cdot & -6C_2L & \cdot & 12C_2 & \cdot & \cdot & \cdot & -6C_2L \\ \cdot & \cdot & -12C_3 & \cdot & 6C_3L & \cdot & \cdot & \cdot & 12C_3 & \cdot & 6C_3L & \cdot \\ \cdot & \cdot & \cdot & -C_4 & \cdot & \cdot & \cdot & \cdot & \cdot & C_4 & \cdot & \cdot \\ \cdot & \cdot & -6C_3L & \cdot & 2C_3L^2 & \cdot & \cdot & \cdot & 6C_3L & \cdot & 4C_3L^2 & \cdot \\ \cdot & 6C_2L & \cdot & \cdot & \cdot & 2C_2L^2 & \cdot & -6C_2L & \cdot & \cdot & \cdot & 4C_2L^2 \end{bmatrix} \quad (5.6)$$

## 5.1 Finite element analysis module

with

$$\begin{aligned} C_1 &= \frac{A \cdot E}{L} \\ C_2 &= \frac{E \cdot I_x}{L^3} \\ C_3 &= \frac{E \cdot I_y}{L^3} \\ C_4 &= \frac{G \cdot J}{L} \end{aligned}$$

For a number of elements within a structure it may be necessary to specify certain end-releases. The term end-release refers to removing the element's stiffness for a certain degree of freedom at one of its ends. For example, a frame element may be required to be pinned at one or both of its ends in order to ensure no moment is being transferred by this element to its supporting member. This pin connection will then be modelled as an end release of the element's rotational degrees of freedom.

Any of an element's degrees of freedom may be released to achieve the desired structural behaviour, on the condition that the structure remains stable. In the case of both ends of a frame being pinned to allow rotation, the member effectively reduces to a truss element with axial stiffness only. The releasing of the degrees of freedom at an element's end is represented by adapting its stiffness matrix. For this formulation, the element's local forces and displacements are partitioned as shown in equation 5.7, where the subscripts  $p$  and  $r$  refer to prescribed and released respectively (Gavin 2012).

$$\begin{Bmatrix} f_p \\ f_r \end{Bmatrix} = \begin{bmatrix} k_{pp} & k_{pr} \\ k_{rp} & k_{rr} \end{bmatrix} \begin{Bmatrix} d_p \\ d_r \end{Bmatrix} \quad (5.7)$$

By simultaneously solving the two rows for the matrix expression in equation 5.7, a single expression may be obtained to represent the new stiffness matrix for only the prescribed degrees of freedom of the element as shown in equation 5.8. The remaining part of the formulation is to fill the rows and columns of the stiffness matrix which corresponds to the released degree of freedom with zeros to indicate that the element has no stiffness to resist the released degree of freedom.

$$\begin{aligned} \{f_p\} &= ([k_{pp}] - [k_{pr}][k_{rr}]^{-1}[k_{rp}]) \{d_p\} \\ \{f_p\} &= [k_{released}] \{d_p\} \end{aligned} \quad (5.8)$$

## 5.1 Finite element analysis module

Using this formulation any selected degree of freedom of an element may be released by adapting its local element stiffness matrix. The same transformation may still be used to transform the element from its local to global orientation. These transformation matrices do however differ for the two- and three-dimensional frame elements considering the additional degrees of freedom. The two-dimensional rotation matrix is very similar to the one used for the truss element and is given in equation 5.9 (Logan 2011, p. 238).

$$[T_{2Dframe}] = \begin{bmatrix} \cos \theta & \sin \theta & 0 & 0 & 0 & 0 \\ -\sin \theta & \cos \theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos \theta & \sin \theta & 0 \\ 0 & 0 & 0 & -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.9)$$

In the three-dimensional case, the rotation matrix consists of the same submatrix,  $[\gamma]$ , placed on its diagonal as shown in equation 5.10. Additional allowance is made to accommodate local axis rotation of an angle,  $\alpha$ . Furthermore, the generic case for rotation becomes undefined in the event where the element is vertical. For this reason, two rotation matrices are defined, one for the rotation of a vertical element and another for any other element orientation. These submatrices for a vertical element and a general element are shown in equations 5.11 and 5.12 respectively (Saouma 1999, eqn 4.41 and 4.49).

$$[T_{3Dframe}] = \begin{bmatrix} [\gamma] & & & \\ & [\gamma] & & \\ & & [\gamma] & \\ & & & [\gamma] \end{bmatrix} \quad (5.10)$$

$$[\gamma_{vertical}] = \begin{bmatrix} 0 & C_y & 0 \\ -C_y \cos \alpha & 0 & \sin \alpha \\ C_y \sin \alpha & 0 & \cos \alpha \end{bmatrix} \quad (5.11)$$

$$[\gamma_{general}] = \begin{bmatrix} C_x & C_y & C_z \\ \frac{-C_x C_y \cos \alpha - C_z \sin \alpha}{C_{xz}} & C_{xz} \cos \alpha & \frac{-C_y C_z \cos \alpha + C_x \sin \alpha}{C_{xz}} \\ \frac{C_x C_y \sin \alpha - C_z \cos \alpha}{C_{xz}} & -C_{xz} \sin \alpha & \frac{C_y C_z \sin \alpha + C_x \cos \alpha}{C_{xz}} \end{bmatrix} \quad (5.12)$$



With

$C_x, C_y, C_z$  as in equation 5.4

$$C_{xz} = \sqrt{C_x^2 + C_z^2}$$

$\alpha$ : The angle of local rotation

An additional requirement for a frame element is to provide a method to mesh it into four sub-elements. This is due to the requirements provided in [SANS 10162-1](#) where the capacity of lateral unsupported elements subjected to bending needs to be calculated. This functionality only applies to elements whose ends are not pinned and simply involves creating three sub-nodes in the element and defining new elements of the same type between these nodes. It is important to note that, if the element has line loads acting on it, these loads must also be correctly divided for each new sub-element. This is needed as the original element and its line loads are removed from the model entirely to avoid doubling the stiffness at the original nodes of the element.

### 5.1.3 Loads

Any structure must be designed to withstand certain loading conditions or a combination thereof. Typical loading examples are self-weight, imposed loadings, wind and snow loads.

The loading can be applied to a structure in various different ways. The first is to simply apply a load of a given magnitude to an existing degree of freedom in a structure. These are known as point loads or moments. The second way is to define a loading on a specific element, these are known as element loads. A third method is to apply a so-called volume load to the entire structure. Volume loads are a good application for gravity loads.

For the FEA module developed in this study, only point and element loads are utilised. Point loads are the only loadings applicable to truss structures since truss elements can not carry bending moments, only axial forces.

---

## 5.1 Finite element analysis module

---

Element loads typically apply to frame elements as they are required to resist loads acting laterally on them, such as floor or wind loads. The element loads are used for frame elements to calculate the element end-forces and end-moments which are used for checking whether or not an element has sufficient capacity to resist the applied loading.

For this implementation, the class, **LoadManager**, is used to contain all the information regarding the loading for a specific model. Classes are also created for point loads, point moments and line element loads. These classes are named **PointLoad**, **PointMoment** and **LineLoad** respectively. For simplicity, the superclass, **Load**, serves as a parent class for any loading which acts at a given point.

A single loading effect, such as wind or gravity, may induce multiple loads. All individual loads induced by a single effect is called a load case. By conforming to the traditional naming the class, **LoadCase**, is utilised to group all the individual loads of a single loading effect together.

Furthermore, **LoadManager** is also equipped to handle a linear combination of load cases. In typical design situations of real world structures there are almost no cases where only one loading condition acts on a structure. For this reason, the class, **LoadCombination**, is utilised as an extension to class **LoadCase** to provide a linear combination of multiple load cases.

For convenience, all the classes associated with the loading of a model are put into a package named “loads”. The class, **LoadException**, is also included to account for errors. The relationship of the “loads” package is shown as a shortened UML diagram in figure 5.8.

## 5.1 Finite element analysis module

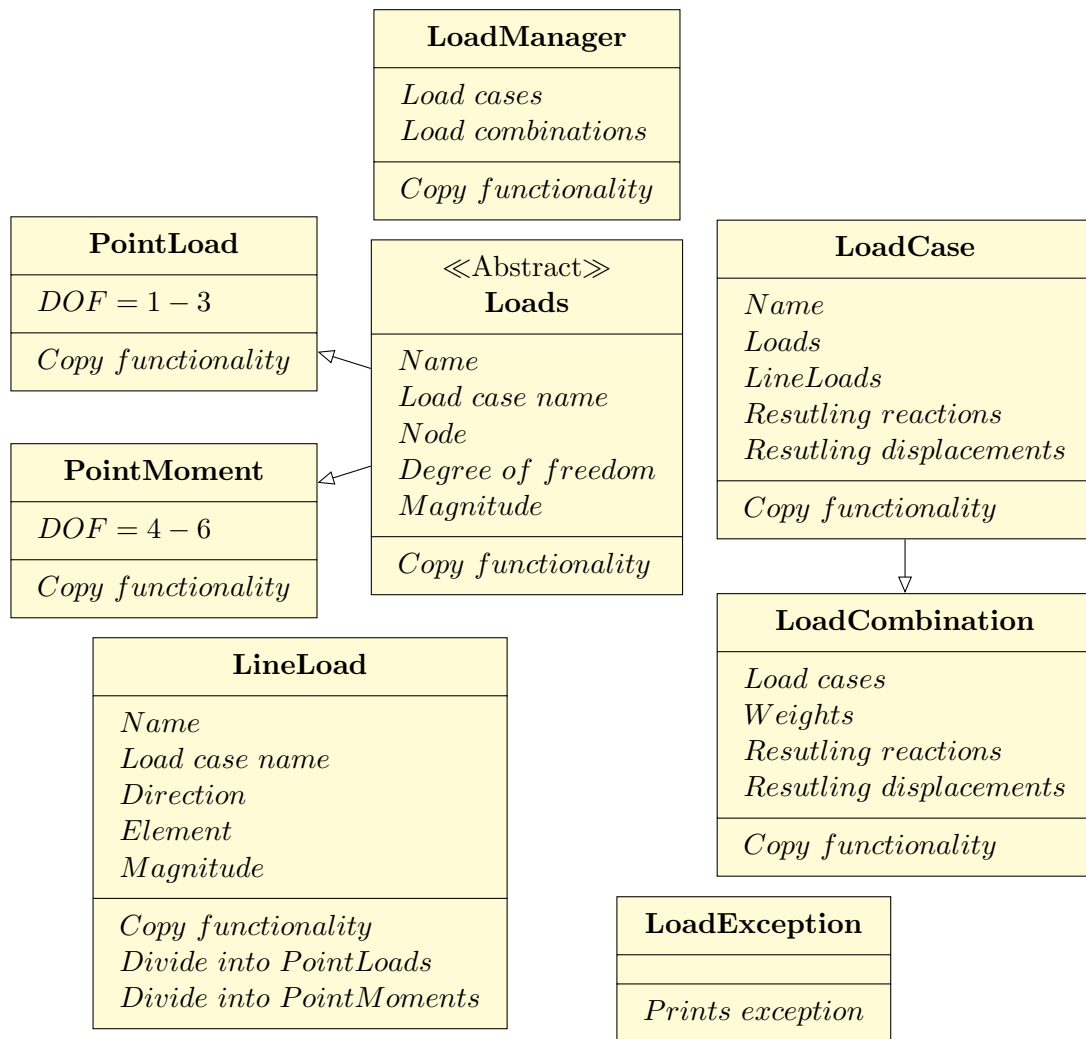


Figure 5.8: UML diagram illustrating the loads package

### 5.1.4 Model

With all the components of the model defined, a functioning object which is compiled from these components must be produced. The class containing the model functionality is named **GenericModel**.

A **GenericModel** has a designated **LoadManager** as well as collections of nodes, elements and supports to describe the model in its entirety. Functionality to add, remove and manipulate these attributes is provided since it will be used by the optimization routine. Further functions of the **GenericModel** class are discussed in more detail in the upcoming subsections.

## 5.1 Finite element analysis module

The model also has the ability to determine its own-weight by adding the own-weight of all the elements together. Weight is typically used as an objective in a structural optimization problem. The simplest way to determine the weight of a structure with the current software architecture is by allocating the calculation to the model which has access to all the elements.

### 5.1.4.1 Perform an analysis

To solve the unknown displacements and forces in a finite element problem, a few standard steps are followed. Firstly, a system of equations is created. These equations comprise of all the element stiffness values, applied loads and prescribed displacement values. These values are acquired from the components present in the `GenericModel` and are sorted according to the degrees of freedom in the system. In other words, all the element values corresponding to a degree of freedom are assembled together. By doing so, the system of equations is created in the form of equation 5.13. The procedure described here is adapted from Cook et al. (2001, p. 40).

$$[K_s]\{D_s\} = \{F_s\} \quad (5.13)$$

The system is then partitioned into its prescribed and free parts which correspond to the supports of the structure. This step is shown in equation 5.14 where the subscripts  $f$  and  $p$  indicate the free and prescribed degrees of freedom.

$$\begin{bmatrix} K_{ff} & K_{fp} \\ K_{pf} & K_{pp} \end{bmatrix} \begin{Bmatrix} D_f \\ D_p \end{Bmatrix} = \begin{Bmatrix} F_f \\ F_p \end{Bmatrix} \quad (5.14)$$

With the partitioned system, the first unknown term to be solved is the unknown displacements,  $\{D_f\}$ . Equation 5.15 shows how this is done.

$$\{D_f\} = [K_{ff}]^{-1} (\{F_f\} - [K_{fp}]\{D_p\}) \quad (5.15)$$

With the unknown displacements calculated, the unknown forces are determined using equation 5.16. This step concludes the solution process.

$$\{F_p\} = [K_{pf}]\{D_f\} + [K_{pp}]\{D_p\} \quad (5.16)$$

For creating, storing, manipulating and solving matrices and vectors, a linear algebra module from the Civil Engineering department of Stellenbosch University is used

## 5.1 Finite element analysis module

---

in this FEA module. This module, named LinearAlgebra, also provides options to solve matrix equations such as equation 5.15. For example, instead of computing the inverse through a computationally expensive operation, a decomposition technique such as Cholesky, LU or LDL may be used. In this study, the LU-decomposition is used to solve equation 5.15 when the matrix,  $[K_{ff}]$ , has less than 850 columns, as it has been shown to be efficient compared to other available methods (Müller 2015). For cases when the  $[K_{ff}]$  matrix contains more than 850 columns, a native method from the MTJ library (Halliday 2015), which is accessible from the provided linear algebra module, is employed for the solution as it performs better than the LU-decomposition for larger matrices (Müller 2015).

An advantage of the finite element method is that allowance is inherently made for the analysis of multiple load cases. Only the force vector,  $\{F_s\}$ , needs to be replaced for each load case, while the rest of the matrices present in equation 5.13 remains unchanged. With this knowledge, the stiffness matrix for a specific structure,  $[K]$ , only needs to be assembled once for all the load cases.

Keeping in mind that the model will only be analysed once, allowance is made to store the resulting displacements and forces for later use. To ensure that the correct results are placed with the correct load case, functionality is provided such that these results can be allocated to each `LoadCase` object directly after it has been obtained. By doing so the element forces and displacements can be assigned based on the preferred load case.

### 5.1.4.2 Copying a model

The process of an optimization routine requires solutions, in this case finite element models, to first be created at random and then from other solutions. Therefore, it is necessary to define a way of creating a copy of a model which can be altered to represent an encoded solution used by an optimization algorithm. For example, during a topology optimization, the original model will have all the variable elements, upon creating a copy to represent an encoded solution, the solution can be decoded and the variable elements which are not present in the decoded solution can be removed from the copied model. By doing so, the copied model is a representation of the solution to the optimization problem, which can be analysed to determine its

## 5.1 Finite element analysis module

---

objective values.

Depending on the nature of the optimization problem, the copied model will have differences in terms of nodal positions, element cross-sections or topology. It is therefore necessary to ensure that the new model is completely independent of the original it was copied from. Another reason for the copy to be independent is for memory purposes. By creating independent copies, the models that are no longer used can be deleted by the Java Garbage Collector to free up space for future models.

The implementation of this functionality is extensive, since each component of the model including nodes, elements, loads and supports must be copied. Therefore, copying methods are introduced to each of these classes. The ability of each class to copy itself simplifies the process of copying an entire model. A new model is assembled by using each attribute of the original model's ability to copy itself and assigning the copy to the new model.

The general procedure for copying a component is outlined in algorithm 1 and the procedure for copying an entire model object is given in algorithm 2.

---

### Algorithm 1 Standard copy pseducode

---

```

1: Create new instance of the object
2: for All defined attributes do
3:   if Attribute must also be copied then
4:     Create a copy of the attribute
5:   end if
6:   Assign attribute to new object
7: end for

```

---

## 5.1 Finite element analysis module

---

### Algorithm 2 Model copy pseudocode

---

```

1: Create new GenericModel instance named model
2: for Node n : all nodes of the current model do
3:   Create a copy of n
4:   Add to the new model
5:   for LoadCase lc : all load cases of the current model do
6:     for Load l : all loads in lc do
7:       if l acts on n then
8:         Add a copy of l to the new model
9:       end if
10:    end for
11:  end for
12:  for Support s : The supports of the current model do
13:    if s acts at n then
14:      Add a copy of s to the new model
15:    end if
16:  end for
17: end for
18: for Element e : all elements of the current model do
19:   Create a copy of e and add it to the new model
20:   for LineLoad ll : The line loads of the current model do
21:     if ll acts on e then
22:       Create a copy of ll (acting on the copy of e) to the new model
23:     end if
24:   end for
25: end for

```

---

#### 5.1.4.3 Meshing a model

To accommodate the meshing of frame elements to determine their bending capacity, the class, **MeshModel**, is implemented as an extension of **GenericModel**. The **MeshModel** introduces the ability to automatically mesh frame elements which are not specified as being pinned on both sides.

The original elements in the model are retained for copying the model. This allows the **MeshModel** to be used by the optimization module described in section 5.2. The main difference between the **MeshModel** and the **GenericModel** is that the elements

---

## 5.1 Finite element analysis module

---

in the `MeshModel` must be meshed before an analysis is performed, otherwise errors will arise.

### 5.1.5 Reporting an analysis

As with any FEA program, results are provided to validate the analysis and design the structure. Although the optimization module does not require a human-readable output, output is required to validate results and ensure that the module functions correctly.

For this reason, a table based output feature is added to list all the components of the model along with the results obtained. The open source Java reporting library `DynamicReports` (Mariaca 2017) is utilised to create these reports. The `DynamicReports` library provides the ability to generate a PDF document containing the output tables, each having its own data and format.

It should be noted that the reporting functionality is only enabled for frame models, both meshed and unmeshed. The truss element models are easily verified by considering the resulting matrices from the analysis. The console printing functionality provided by the `LinearAlgebra` module is used for verifying truss structures. However, this is not an option for frame structures as they have more degrees of freedom and the force vectors contain significantly more entries.

A small example analysis report from the structure shown in figure 5.9 is included in appendix C. This structure is a two-dimensional frame structure with only two elements. For the analysis, each element has been divided into four sub-elements to illustrate how a meshed model containing sub-nodes and sub-elements is reported in the output.



## 5.1 Finite element analysis module

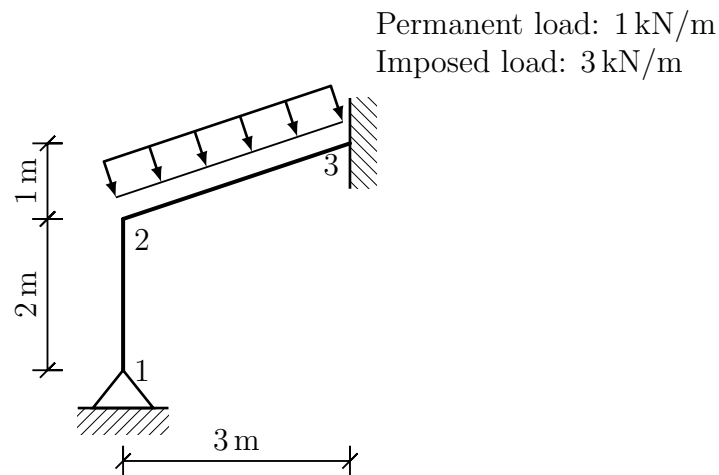


Figure 5.9: Structure to demonstrate analysis report

### 5.1.6 A note on units

It is well known that the choice of units and using the same units throughout are of great importance when it comes to FEA. In this module, no units are specified. This choice is made with the reasoning that several optimization problems may be defined in a different set of units, which would mean that this module cannot be used without performing tedious unit conversions.

It is however recommended that the base SI (International System of Units) units are used. This ensures unit consistency throughout the modelling process. An option is provided in the FEA module to write results in engineering notation which formats any value to be in the form  $10^x$  with  $x$  being a multiple of three. This option simplifies the results interpretation in terms of forces and stresses which is typically measured in kN and MPa.

The database functionality implemented for reading cross-section profiles, discussed in section 5.1.1, has been adapted to convert measurements to standard SI meters. The original values are assumed to be in millimeters as in the Southern African Institute of Steel Construction (2013). However the units can be changed by altering the source code of the `Profile` subclasses.

## 5.2 Optimization module

Various methods to solve an optimization problem were identified in chapter 4. In this study, focus is now placed on the genetic algorithm with elitism (GA) for single-objective problems and the non-dominated sorting genetic algorithm (NSGA-II) for multi-objective problems. The decision to use the GA and NSGA-II is made because it has been proven to yield good results for structural optimization problems (Rahami et al. 2008; Tang et al. 2005; Barraza et al. 2017; Vo-Duy et al. 2017; Koumar et al. 2017) and its inherent ability to cater for different variable encodings as well as its ease of implementation. In the remainder of this section GA refers to both GA and NSGA-II.

By implementing various encodings, different parts of a solution can easily be distinguished by considering the variable type. This distinction is advantageous for the structural optimization problem where two or more different aspects are considered, for example, the size, shape and topology of a structure. The respective variables associated with each aspect can be identified and the evolutionary operations of the GA can be applied to each type of variable. Another advantage is that this distinction simplifies the process of decoding the solution to retrieve the structure it represents.

Without the chosen distinction between the different variables, additional bookkeeping would be required to remember which variables are associated with which aspect of the structure. Without such a clear distinction of variables, the implementation may be prone to errors and may result in additional computation within each iteration of the optimization routine.

The only real disadvantage of using different types of variables is that each type requires its own strategies for how operations are performed. Referring to the crossover operation utilised by the GA described in section 4.1.1, this operation differs for a binary and real-value variable, discussed in sections 4.1.2 and 4.1.3 respectively. Therefore, it should be explicitly be defined for both.

As different optimization software libraries have already been developed, it is suitable to use one of these libraries for the implementation as opposed to implementing an

entire optimization module from scratch. Most of the available libraries have been developed over a number of years and it would be impossible to develop a module that matches the standard of the available libraries. It would be advantageous to adopt one of the libraries and adapt it to suit the current structural optimization problems.

From the wide range of available optimization libraries, only those written in Java are considered to allow for easy integration with the FEA module. A few of the available libraries are listed below:

- JGAP (Steghoefer 2015)
- Jenetics (Wilhelmstötter 2016)
- MOEA Framework (Hadka 2015)
- Watchmaker Framework (Dyer 2010)

From the available libraries, the MOEA Framework is used in this module. The reason this library was chosen over the other available options is that it is still actively supported with updates and bug fixes. The other libraries are no longer maintained as their development has been halted. A brief overview of the MOEA framework is provided in the next subsection.

### **5.2.1 The MOEA Framework, a brief overview**

The MOEA (multi-objective evolutionary algorithm) Framework is an open source Java library for multi-objective optimization problems with an extension that accommodates problems of a single-objective nature. The library's development started in 2009 and since then a number of improvements and extensions have been made to create a very powerful optimization tool. It provides an extensive set of features including: monitoring performance, adding custom algorithms, adding new variable types and displaying results. The design of the library is modular and simple to follow, therefore, allowing for adaptation to suit structural optimization problems.

The MOEA Framework also includes several algorithms other than the GA without elitism and NSGA-II which can be used including: differential evolution, particle swarm optimization and genetic programming. The general architecture of the

framework is designed for easy extension to include more algorithms, problems and variation operators. The addition of any class to represent a new feature, such as an algorithm, problem or variation operator, needs to extend the appropriate superclass. For example, to define a new algorithm the class must extend the framework's **Algorithm** superclass. The framework also provides the ability to change the parameters of an algorithm, such as population size, before executing an optimization.

Considering that the GA with the elitism strategy, which is discussed in section 4.1.1, is to be used in this study and that it is not natively supplied by the MOEA Framework, it must be added. This is done by creating an appropriate class, named **GAElitism**, which extends the MOEA Framework's **Algorithm** superclass.

The evolutionary operators such as crossover and mutation are all grouped under the class, **Variation**. This standardises the use of these operators and allows for the addition of new ones. The framework does have native implementations of a number of popular variations including crossover and mutation for both binary and real-value variables.

For the encoding of solutions, the class, **EncodingUtils**, is provided which creates binary, boolean and real-value variables. This class simplifies the effort for deciding on an encoding scheme and allows for the reallocation of time to other parts of the optimization. It is important to note that solutions must be decoded in order to perform the FEA. Since a **GenericModel** must be created from each solution, a function needs to be implemented in the adapted MOEA Framework to suit the current structural optimization application.

Another important feature of the library is the native ability to compare and sort solutions according to their non-dominating objective values. A pareto optimal set of solutions is obtained allowing for easy interpretation of the result. It is also possible to manually specify how solutions should be compared, but for this study the native methods are deemed sufficient.

By using the means provided by the MOEA Framework to encode and decode variables, perform variation operations and specify optimization algorithms as well

as their parameters, an excellent basis for adapting an existing software module to perform optimization routines is established. The next step is to adapt the existing architecture to suit the MOEA framework specifications. In the following subsection, it is discussed how the MOEA framework is used to create an optimization framework specifically for structural optimization applications.

### 5.2.2 Adaptation to structural optimization

With the availability of the MOEA Framework to perform optimizations, adaptations are made to merge the features of the MOEA Framework with the existing structural analysis implementation, discussed in section 5.1. The `GenericModel` class functions as a solution representation of the structural optimization problems considered in this study, therefore, a method to encode it into a solution which can be used by the MOEA framework must be established.

Furthermore, the structural optimization problems, namely, size, shape and topology, are defined to suit the architecture of the MOEA Framework. Classes to define objectives and constraints to the structural problems and classes to define the specifics of these problems are added. The details of these additions are described in the upcoming subsections.

#### 5.2.2.1 Objectives and constraints

Any optimization problem consists of at least one objective and may be subjected to a number of constraints. Therefore, objectives and constraints must be contained in any problem representation as attributes. It is deemed efficient to provide these attributes in the superclass of any problem such that all the subclasses inherently contain these attributes.

According to the MOEA Framework, both constraints and objectives are attributes of the `Solution` class. This enables each solution the ability to state whether or not it satisfies the given constraints and its calculated objective values. These objective and constraint values are used for comparing different solutions to sort the population.

It is important to note that there may be a number of objectives and constraints present in a given problem. The MOEA Framework makes provision for this as each `Solution` object stores two numerical arrays, one for the objective and another for

the constraint values. When a solution is evaluated, the numerical values of the objectives and constraints are calculated and stored in their respective arrays.

Adapting to the existing architecture, two interfaces, `ConstraintHandler` and `ObjectiveHandler`, are used to define any specific constraint or objective. These interfaces are parametrised to comply with the solution representation, which in this case will always be a `GenericModel`. This parametrisation enables the constraints and objectives to be used for any future problem definition such as a scheduling problem.

The above-mentioned interfaces prescribe the ability to compute and assign the values of the objectives and constraints. For the objectives, a weighting function is provided, although, it is only applicable when a linear combination of objectives is considered. In this study, the classes, `DisplacementConstraintHandler` and `MassObjectiveHandler`, implement these interfaces.

To accommodate the handling of multiple objectives and constraints, the classes, `CompoundConstraintHandler` and `CompoundObjectiveHandler`, are used and implement the aforementioned interfaces. These classes simply allow for multiple handlers to be added and automatically combine the handlers' calculate functions.

Figure 5.10 shows a shortened UML diagram which illustrates how these objective and constraint handling interfaces function. The aforementioned classes which implement these two interfaces are also shown in the figure.

These created interfaces and classes which implement them can be used to define the objectives and constraints of any problem. The compounding classes enable the use of more than one objective or constraint. This is essential as practical structural optimization problems have a number of objectives, for example, displacement and weight minimization. In the current implementation, allowance is made for stress and displacement constraints and also for mass and displacement objectives.

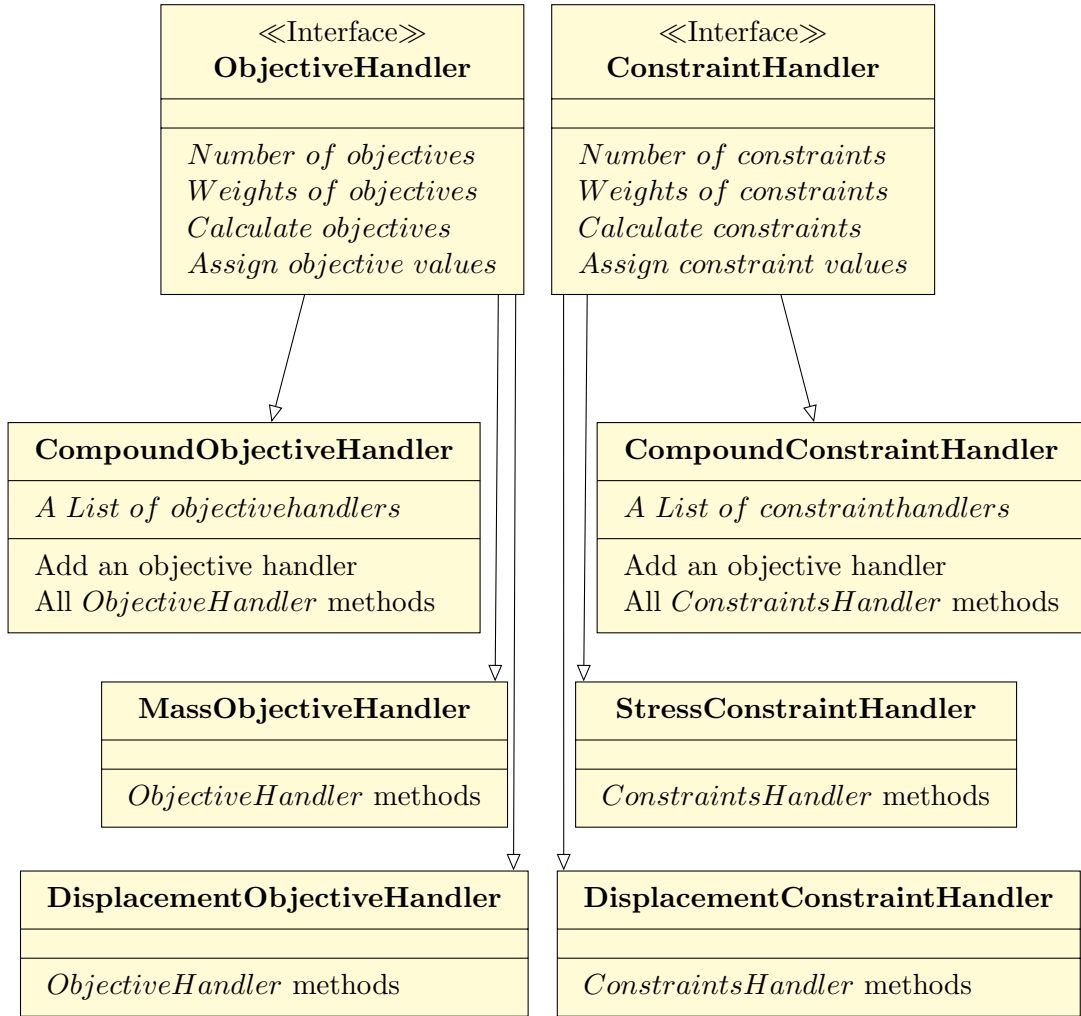


Figure 5.10: UML diagram illustrating the constraint and objective handlers

### 5.2.2.2 Basic Problem definition

With methods to handle both objectives and constraints established, the problem definition can be discussed. The class, **AbstractStructuralProblem**, is the first under discussion. This class is an extension of the MOEA Framework's **AbstractProblem** class and serves as the basis for all implementations of structural optimization problems. This class is later extended to define size, shape, topology and simultaneous problems.

The **AbstractStructuralProblem** class is parametrised to specify which object class serves as the representation of a solution to the problem. The same parameter type is assigned to its objective and constraint handlers, which are both of the compounding

type as previously described in section 5.2.2.1.

As a final specification, the **AbstractStructuralProblem** class prescribes that methods are defined by which a **Solution** object, from the MOEA Framework, can be created from the parametrised object which represents a solution of the optimization problem. In other words, the **AbstractStructuralProblem** class prescribes that any class extending it must provide methods to encode and decode a **Solution** object from the MOEA Framework to an object of the parametrised type. This functionality is required as solutions must be decoded in order to analyse the candidate structure for determining its objective and constraint values and encoded to be used by the optimization routine to perform variation operations such as crossover.

In this study, the **GenericModel** class represents a solution to an optimization problem. This is because a **GenericModel** is used to calculate the required objective and constraint values for the structural problems. For this reason, the class, **StructuralProblem**, is established and extends the **AbstractStructuralProblem** class with the **GenericModel** as parametrisation. The **StructuralProblem** class implements the method to evaluate a solution, while the encoding and decoding operations are left for problem specific extensions of this class. This is done because the method for encoding a size optimization solution, for example, differs from the encoding of a topology optimization solution. The pseudocode for the evaluate method is shown in algorithm 3.

---

**Algorithm 3** **GenericModel** evaluation pseudocode

---

```

1: Build a GenericModel from the Solution
2: if Model must be meshed then
3:   Mesh the unpinned frame elements in the model
4: end if
5: Analyse the model
6: if The analysis was successful then
7:   Calculate and assign objective and constraint values to the Solution
8: else
9:   Assign maximum objective and constraint values, indicating a bad solution
10: end if

```

---



### 5.2.2.3 Size problem definition

To find solutions to size optimization problems, the class, `SizeProblem`, is implemented. The `SizeProblem` class extends `StructuralProblem`, described in section 5.2.2.2, and provides the ability to decode a `Solution` object from the MOEA Framework to a `GenericModel` and vice-versa. These abilities are briefly described in this section.

As the variables of a size problem are chosen from an available list of cross-sections, an integer binary encoding is used. These integer variables range from zero up to, but not including, the number of available cross-sections for a specific member. The integer value of the variable corresponds to the index of the selected cross-section in the array of available cross-sections. In other words, the binary variable represents the index of the chosen cross-section in the array of available cross-sections. By doing so, one array of cross-sections may be kept for each element while only the binary integers are used in the optimization.

By using binary integer variables, standard methods for the crossover and mutation operations, which are included in the MOEA Framework, are used. This not only avoids implementing custom operators, but also simplifies the required methods for decoding a solution back to a `GenericModel`.

For the specification of element groupings, the class, `SizeSettings`, is used. This class allows for specifying the selection of cross-sections available to each variable member and which elements are designated to have the same cross-section as this member.

In summary, the `SizeProblem` class is used to define a size optimization problem. This is done by establishing a means to encode and decode size variables as well as member groupings within the structure. To optimize the size of a structure, the elements which are considered to be variables, their list of available cross-sections and member grouping must be defined.

### 5.2.2.4 Topology problem definition

For the implementation of a topology optimization problem, the class, `TopologyProblem`, is implemented. Similar to class `SizeProblem`, `TopologyProblem`

also extends the **StructuralProblem** class. The approach followed by this implementation is described below.

A topology optimization problem aims to determine the presence or absence of elements within a structure. Since the only options for a variable member is to either be present or not, a boolean variable is well suited for such a problem. A boolean variable implementation is provided by the MOEA Framework in the form of a single binary string, which can have a value of either one, or zero.

With only boolean variables present, the encoding and decoding of a topology optimization problem is elementary. Each element that is selected to be a variable is allowed to have its original cross-section in the case it is present in the structure, and a “zero” cross-section if it is left out.

The term “zero” cross-section refers to a cross-section which causes an element to have no stiffness or weight contributions to the structure. This means setting the cross-sectional area and moment of inertia properties of the cross-section to zero. By doing so, the element is effectively removed from the structure, although it still exists in the model to be reinstated during another iteration.

To accommodate grouping and the variable selection functionality, the class, **TopologySettings**, is used. This class keeps track of the elements considered as variables and also the elements that are grouped together. By doing so, the problem details and general definition are separated. This enables the ability to handle unique structures without changing the problem definition.

The **TopologyProblem** class is used to define a topology optimization problem. This problem determines whether or not an element should be present or removed from a structure. To optimize a structure’s topology, the removable members must be identified and the member grouping to determine which elements are dependent on others.

#### 5.2.2.5 Shape problem definition

A shape optimization problem deals with the nodal positioning of a given structure. To cater for such problems the class, **ShapeProblem**, is used.

The variables of a shape optimization problem are continuous, with minimum and maximum bounds specified for each node. For these variables, the MOEA Framework provides real-value variables, where upper and lower bounds are specified upon instantiation. This available variable type simplifies the implementation and allows for simple encoding and decoding methods.

Similar to the other optimization approach classes, size and topology, the **ShapeProblem** class is accompanied by the settings class, **ShapeSettings**. This class allows for bookkeeping of which nodes can be moved and which must be cloned from the variables. Furthermore, the settings class keeps track of each encoded variable and its associated nodal position.

Since symmetry is prescribed for a number of structures, the **ShapeSettings** class is extended to clone a selection of the variable node characteristics. Cloning refers to copying one or more nodal positions, x, y or z, from one node to another. This cloning process may be set to clone a value and make it negative to cater for the case where the line of symmetry lies on the origin.

It is interesting to note that for a shape problem, each directional coordinate of a node can be regarded as an independent variable. For instance, each directional coordinate, x, y and z, can be assigned an upper and lower bound which yields three variables. This means that if all the directional coordinates of all the nodes within a three-dimensional structure are allowed to be variables, the total number of variables will be three times greater than the number of nodes in the structure.

To summarize, the **ShapeProblem** class defines how the shape of a structure is optimized. This includes a feature to accommodate symmetric nodes as well as independent directional variables of each node. This is required as in a number of problems only the x-coordinate is altered while the y- and z-coordinates are chosen to remain in place.

#### 5.2.2.6 Combination problem definition

It is possible to consider more than one structural aspect, size, topology or shape, within an optimization problem. This may either be simultaneous or sequential.

However, these problems have increased complexity as not only more variables are introduced, but also the relationship between these variables may conflict. For example, if shape optimization positions a node to improve the structural performance, while topology optimization removes all the elements connected at that node, the nodal positioning of the shape optimization becomes redundant.

These combination problems are created using the same principles as the individual problems discussed in sections [5.2.2.3](#) to [5.2.2.5](#). Furthermore, the same settings classes are used for a combination problem. For a combination problem, the only required task is to combine the encoding and decoding methodologies from the respective problems.

It is important to note that each structural aspect uses a different variable type. Size uses binary integers, topology uses boolean and shape uses real-value variables, enabling easy identification. By using the variable type to identify different variables, the encoding and decoding methodologies from the individual problems are separated. These different encodings enable the variables associated with each aspect to be identified and encoded or decoded in a manner similar to the individual problems.

In this implementation the following combination problems are defined and compared to one-another for various truss structures in chapter [6](#).

- Simultaneous size, shape and topology
- Sequential topology and size
- Sequential topology, size and shape
- Sequential size, topology and shape

### **5.2.3 Reporting an optimization**

Reporting functionality is added to the optimization module to provide feedback on the routine. The reported information is listed below and an example report is included in appendix [D](#).

- The variable elements and their available cross-sections

- Variable grouping
- The time the optimization used to complete

These reports provide a method for comparing the performance and results between different optimizations of the same structure. The results may be influenced by using differing population sizes and maximum number of iterations. From these comparisons the optimization parameters can be calibrated. Reports may also be kept as a record of the results from multiple optimizations.

## 5.3 Visualization module

To create finite element models and visualize structural behaviour, a visual representation of a structure is required. By using a three-dimensional computer generated representation of the created model, input errors and software bugs can easily be identified. In terms of optimization, it is also preferential to specify optimization settings in a visual manner as opposed to a text-based manner. For this reason, a visualization module is added to provide the means for inspecting and evaluating a FEA model and optimization inputs.

Considering that both two- and three-dimensional problems are catered for in the FEA and optimization modules, the visualization of both two- and three-dimensional structures are accommodated. Two-dimensional structures are visualized as three-dimensional structures without their third dimension. This allows the visualization module to only be designed for three-dimensional structures. As the two- and three-dimensional FEA elements discussed in section 5.1.2 are defined to have their y-axis vertical, the y-axis of the visualization is also taken as vertical.

The visualization is rendered using standard JavaFX and the three-dimensional library FXyz (Pereda et al. 2016). These libraries enable rendering of three-dimensional extrusions within an environment which includes pan, rotate and zoom capabilities.

To successfully render structural elements, each **Profile** class is given a method to provide the polygon points of the cross-section. These points are used to draw the cross-section, while the nodes of the structural elements provide the spatial positions of the members. Any object representation of a member consisting of a cross-section

### 5.3 Visualization module

attribute which is an extension of the `Profile` class can be drawn. For members without a `Profile` cross-section, a square box is used by default to represent these members.

An arrow shape is created by combining a cylinder and cone object. This arrow shape enables the visualization of loads. Currently only point and line loads are available in the visualizer. These two load types are deemed to be sufficient for viewing purposes as structures are typically only subjected to these loads directly. Bending moments are in turn induced by these loads but these moments do not need to be visually represented.

By using the viewer, the model representation is easily visualized and errors due to incorrect input are prevented. An example of how the viewer is used to render a portal frame is shown in figure 5.11.

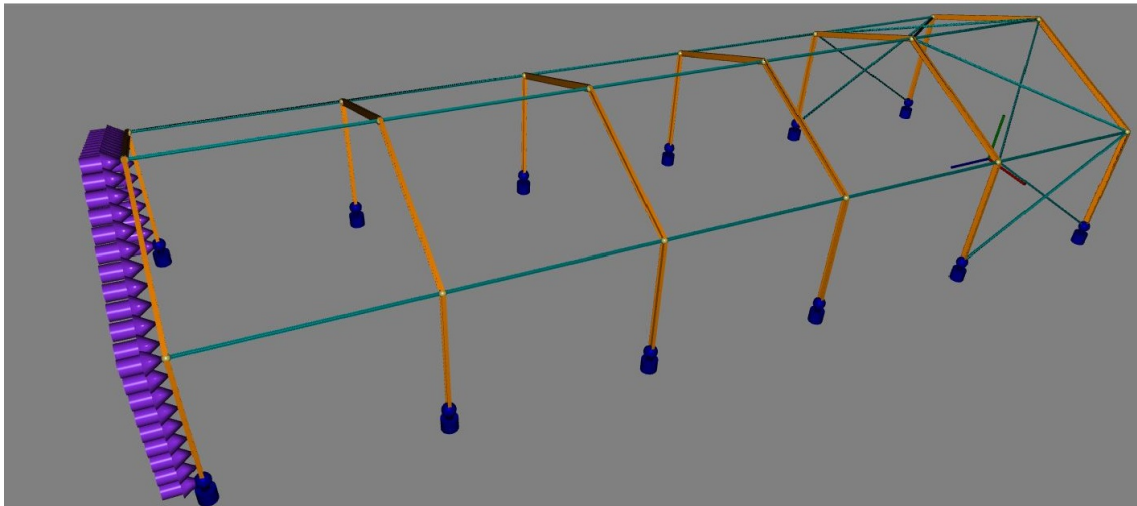


Figure 5.11: An example of the visualization module

The visual input also allows for the specification of multiple load cases and load combinations. This avoids the process of specifying such values in a text format. A disadvantage of text based input is that it is error prone due to the inability to visually perceive how data is inserted. In the visualizer, each load combination can be seen independently ensuring that the data is correctly inserted.

An extension to the visualization module is the ability to perform a FEA or optimization. Both of these produce their respective report files which can be

### 5.3 Visualization module

---

examined upon completion of the FEA or optimization. The parameters required for each of these routines, such as whether or not to mesh elements for the FEA or the population size for the optimization, can be specified.

As a final remark, the Java language provides a **Serialization** interface which is used to save any model created in the visualization to an external file. These files are imported to the visualization module to resume editing or to execute a finite element analysis or optimization of a model.

## 6. Single-objective truss optimization technique comparative study

Many studies have been done to find an effective algorithm for optimizing structures (Coello et al. 1994; Camp and Bichon 2004; Tang et al. 2005; Lamberti 2008; Kaveh and Talatahari 2009a; Luh et al. 2011; Jalili et al. 2015; Mortazavi et al. 2016), while little research has been done to quantify the improvement of the resulting structure if a more comprehensive optimization approach is used. In other words the question “How much is to gain by optimizing the size, shape and topology of a structure as opposed to just the size?” is raised. It is well-known that considering the size, shape and topology of the structure simultaneously will produce the best result (Luh et al. 2011; Miguel et al. 2013), although this is rarely used due to the additional complexity. This study aims to justify the usage of a more complex approach in favour of a significant improvement in the resulting structure.

Comparisons between optimization approaches have previously been made. For example, Kocvara et al. (1996) present results by comparing a topology and size problem with a topology, size and shape problem and Achtziger (2007) compared the simultaneous with the staged approaches. The current study differs from others in the way the comparison is presented. Neither of the aforementioned studies considered the increased computation for more complex approaches, or a comprehensive set of approaches as done in this study.

In an attempt to quantify the improvement of the resulting structure, the optimization framework, described in chapter 5, is utilized for optimizing various benchmark truss structures that are found in literature. These structures include both two- and three-dimensional trusses.

To achieve an extensive range of tests, seven different routines are defined for testing and comparison. These include the three individual approaches, size, shape and topology, along with three staged routines and a simultaneous approach. The staged and simultaneous approaches are:



- 
1. Topology followed by size optimization (TS)
  2. Size, followed by topology and then shape optimization (STS)
  3. Topology optimization, followed by shape and then size optimization (TSS)
  4. Simultaneous optimization where size, shape and topology are considered at the same time (SIM)

The majority of the optimization problems found in literature have only one objective which is weight minimization (Achtziger 2007; Kaveh and Talatahari 2009b; Kaveh 2013; Mortazavi et al. 2016). For this single-objective problem, the GA with elitism which is added to the MOEA Framework in the optimization module, described in section 5.2, is used. The parameters used for this GA are outlined in table 6.1. These parameters were obtained by attempting a number of different combinations and selecting suitable values which yielded good results.

To obtain reliable results, ten independent runs were executed. From these runs the average time and best resulting structure are presented as results. Multiple runs are required as the result obtained from a meta-heuristic search algorithm may deviate for each run.

For staged optimization, namely the TS, STS and TSS approaches, the number of iterations are divided to allow an acceptable amount for each stage. The transition from one stage to the next is defined as taking the best solution from the previous stage as a template for the next stage. For example, if a size routine must succeed a topology routine, the size routine will use the best topology found by the topology optimization routine and generate a new population by randomly initialising the cross-sections for the specific truss.

Table 6.1: Parameters used for the GA

Parameter	Value
Population size	80
Total iterations	1000
Elite solutions	5

## 6.1 10-Bar truss

One of the more popular structures, typically used as a starting point for evaluating new optimization algorithms, is the 10-Bar truss. This structure was first used by Schmit (1974) and consists of ten elements connected by six nodes as shown in figure 6.1. The design parameters used for this problem are listed in table 6.2. The size variables are selected from a discrete set of 42 cross-sections ranging from  $1045 \text{ mm}^2$  to  $21\,613 \text{ mm}^2$  as shown in table B.1 in appendix B.

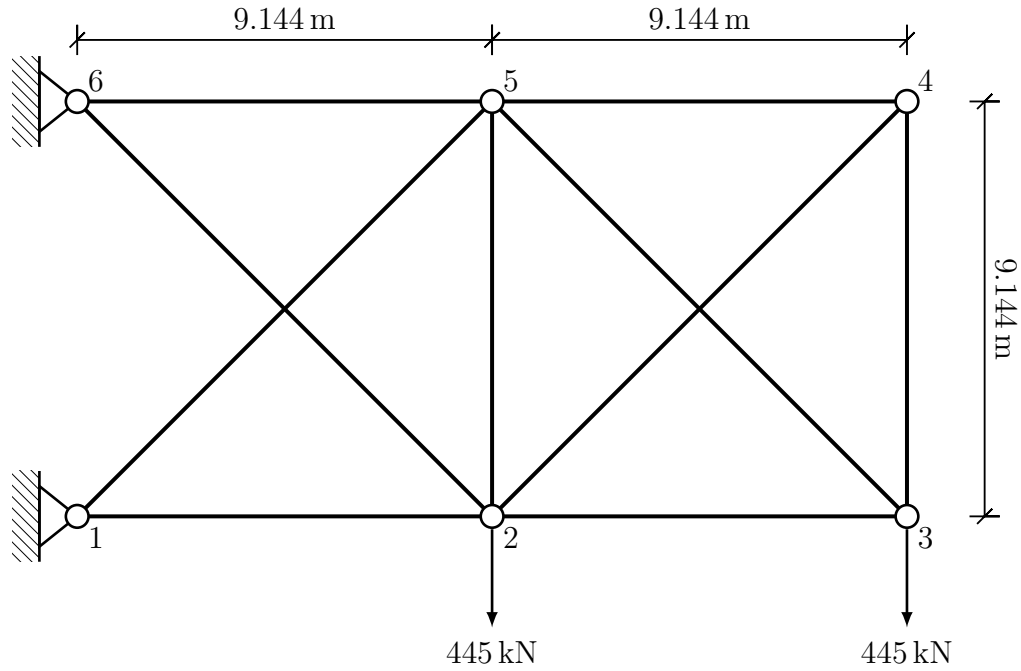


Figure 6.1: 10-Bar truss

Table 6.2: 10-Bar truss design parameters

Parameter	Value
Young's modulus	68.95 GPa
Material density	$2768 \text{ kg/m}^3$
Allowable compressive stress	172.25 MPa
Allowable tensile stress	172.25 MPa
Allowable displacement	50.8 mm

## 6.1 10-Bar truss

For this optimization problem, the selection of variables is simple. All the elements are regarded as size and topology variables. For the shape optimization approach, the nodes on the bottom chord of the truss cannot move, while the nodes on the top chord can move in the vertical direction as defined in equation 6.1. This results in the problem consisting of ten size and topology variables with three shape variables.

$$\begin{aligned} 5 \text{ m} &\leq y_4 \leq 25 \text{ m} \\ 5 \text{ m} &\leq y_5 \leq 25 \text{ m} \\ 5 \text{ m} &\leq y_6 \leq 25 \text{ m} \end{aligned} \tag{6.1}$$

The results of the various optimization approaches are shown in table 6.3. The execution time and the percentage reduction from the base structure are also indicated. The weight of the base structure is determined from assigning the largest cross-section to all the members. This weight is calculated as 6367 kg.

Table 6.3: 10-Bar truss results

Approach	Time (s)	Result (kg)	Reduction (%)
Size	1.50	2490.56	60.9
Topology	1.02	3735.37	41.3
Shape	1.16	5365.77	15.7
TS	1.20	2507.98	60.7
STS	1.31	2305.54	63.8
TSS	1.23	2383.88	62.6
SIM	2.37	1230.24	80.7

To prove the adequacy of the GA used, the results obtained are compared to those found in literature. For the size problem, the resulting weight of 2491 kg is comparable to the 2540 kg of Sivakumar et al. (2004) and the 2474 kg of Nanakorn et al. (2001). For the SIM approach, the GA's result of 1230 kg compares well to those of 1282 kg and 1235 kg obtained by Tang et al. (2005) and Rahami et al. (2008) respectively.

These comparisons indicate that the GA provides reasonable results. Therefore, the algorithm can be regarded as a suitable optimization routine making it eligible for this comparative study. It is also important to ensure consistency throughout this

comparative study. This is done by using the the same algorithm for all test problems.

The resulting optimal structure from the SIM approach is shown in figure 6.2. The figure shows the optimal structure's topology along with how the nodes are moved in order to produce the lightest structure. Since no elements are connected at node 4, it has been removed.

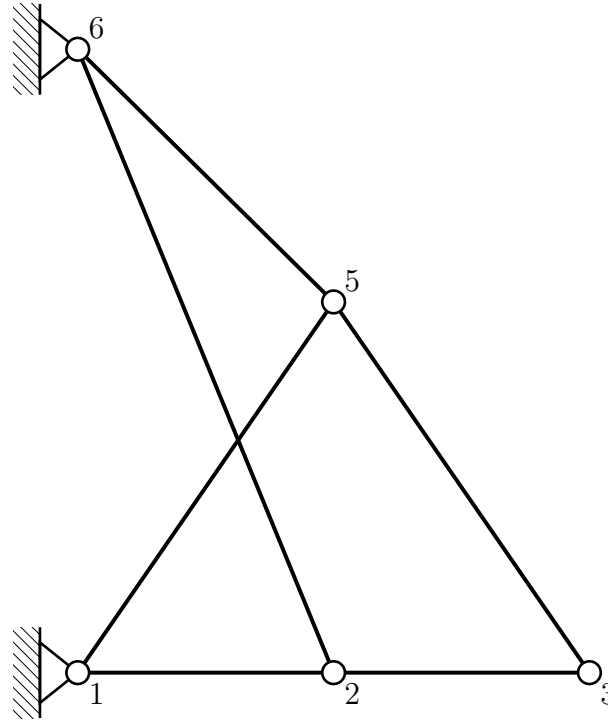


Figure 6.2: 10-Bar truss simultaneous optimization result

The performance of the various approaches with respect to weight versus iteration is illustrated in figures 6.3 and 6.4 by plotting the current best solution for each iteration. The performance data is presented in two figures due to the difference in nature between the routines. The size and SIM routines converge in significantly less iterations, therefore, different scales are used on the horizontal axis of these figures. These differences may be attributed to the fact that the staged routines only proceed to the next stage after a certain number of iterations. The performance of the GA can be seen in more detail in figure 6.3. The maximum number of iterations is shown in figure 6.4 to illustrate what happens when the transition is made from one stage to another during the execution of the respective routines. These transitions may be observed at either 400, 600 or 800 iterations.

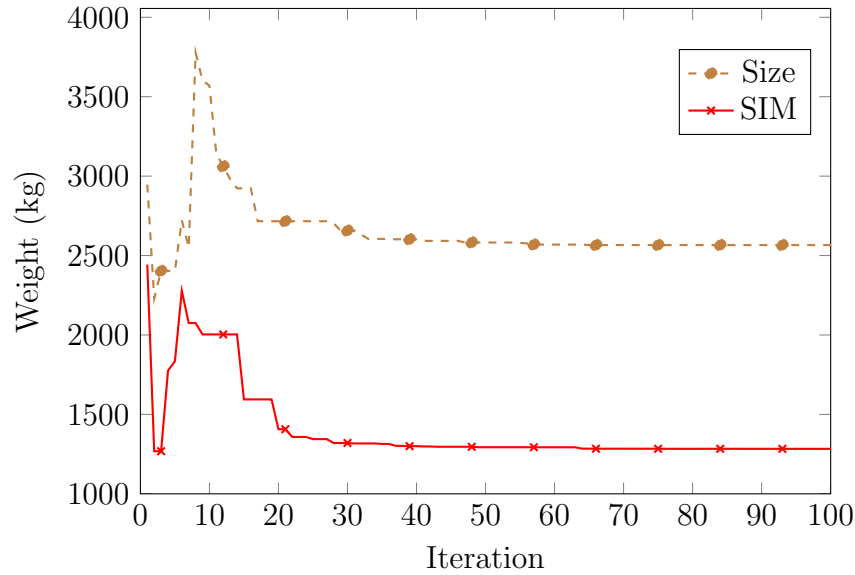


Figure 6.3: Performance of the size and SIM approaches for the 10-Bar truss

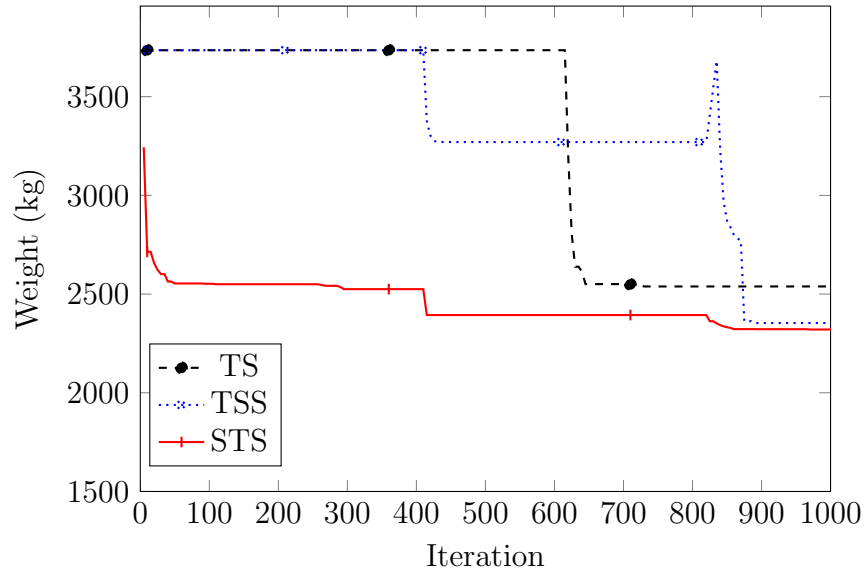


Figure 6.4: Performance of the TS, STS and TSS approaches for the 10-Bar truss

As expected, the SIM optimization routine produces the lightest structure. This is clear in figures 6.3 and 6.4. However, it is interesting to note that the standalone size optimization performs better when compared with two of the staged approaches. The reduction percentage from the staged optimization routines is only a 3% improvement to the size approach. The performance of the staged approaches may be improved by introducing more alterations between aspects as frequently found

in literature (Achtziger 2007). For example, a better result may be obtained by considering several STS routines in succession. This type of routine will, however, require more iterations or a reduction in the number of iterations allocated to each stage.

The topology and shape optimization routines are not shown due to their relatively poor performance. From these results the initial conclusion can be made that the shape and topology optimization routines do not perform well as single approaches. However, they do allow for improvement when used in conjunction with other strategies.

The weak performance of the shape and topology approaches may be attributed to their respective limitations. For example, topology optimization may only remove elements in the structure. In the case of the structure only having 10 elements, the number of elements that can be removed before the structure becomes unstable is very small. This limitation may become insignificant in more complicated structures. A similar argument can be made for the shape optimization approach where the nodes that can vary in coordinates will only reduce the weight if the length of elements are reduced. As a number of nodes have predefined constraints, the effectiveness of this approach is quite limited.

The behaviour of the TSS routine is interesting in this problem. During the transition from shape to size optimization at the 800<sup>th</sup> iteration, the random initialization of the size variables causes an increase in the weight of the structure. This weight is then reduced to produce a good end result by the size stage of the optimization.

## 6.2 25-Bar truss

The first three-dimensional structure investigated is the 25-Bar space truss shown in figure 6.5. The problem definition was taken from Schmit (1974) with the nodal coordinates listed in table 6.4 and the design parameters listed in table 6.7. The element information along with the grouping of elements are shown in table 6.5 and the loading conditions applied to the structure is shown in table 6.6. The 30 available cross-sections are defined in table B.2 in appendix B.

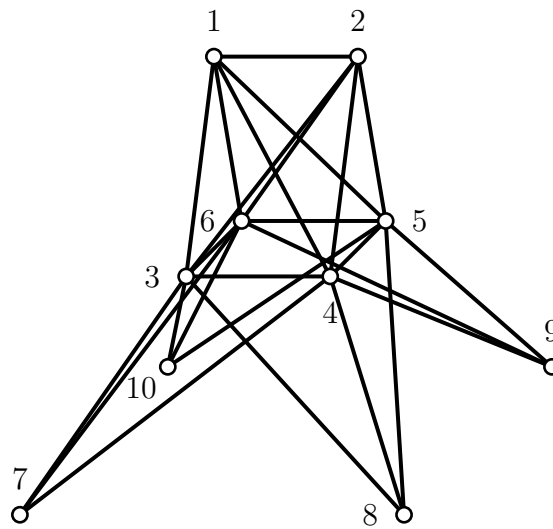


Figure 6.5: 25-Bar truss

Table 6.4: 25-Bar truss nodal coordinates

Node	x (m)	y (m)	z (m)
1	-0.9525	0.0	5.08
2	0.9525	0.0	5.08
3	-0.9525	0.9525	2.54
4	0.9525	0.9525	2.54
5	0.9525	-0.9525	2.54
6	-0.9525	-0.9525	2.54
7	-2.54	2.54	0.0
8	2.54	2.54	0.0
9	2.54	-2.54	0.0
10	-2.54	-2.54	0.0

Table 6.5: 25-Bar truss element information

Group	Element name (end nodes)
A1	1(1,2)
A2	2(1,4), 3(2,3), 4(1,5), 5(2,6)
A3	6(2,5), 7(2,4), 8(1,3), 9(1,6)
A4	10(3,6), 11(4,5)
A5	12(3,4), 13(5,6)
A6	14(3,10), 15(6,7), 16(4,9), 17(5,8)
A7	18(3,8), 19(4,7), 20(6,9), 21(5,10)
A8	22(3,7), 23(4,8), 24(5,9), 25(6,10)

Table 6.6: 25-Bar truss loading information

Node	$F_x$ (kN)	$F_y$ (kN)	$F_z$ (kN)
1	4.4482	-44.4822	-44.4822
2	0	-44.4822	-44.4822
3	2.2241	0	0
6	2.6689	0	0

Table 6.7: 25-Bar truss design parameters

Parameter	Value
Young's modulus	68.9 GPa
Material density	2768 kg/m <sup>3</sup>
Allowable compressive stress	275.79 MPa
Allowable tensile stress	275.79 MPa
Allowable displacement	8.89 mm

Only a few nodes form part of the five shape variables. Furthermore, grouping is used to reduce the amount of size and topology variables to eight. These decisions force the structure to stay symmetrical. The detail regarding shape variables is shown in table 6.8.



Table 6.8: 25-Bar truss variable detail

Variable	Detail
Shape Variables (mm)	$0.508 \leq x_4 \leq 1.524$
	$1.016 \leq y_4 \leq 2.032$
	$2.286 \leq z_4 \leq 3.302$
	$1.016 \leq x_8 \leq 2.032$
	$2.54 \leq y_8 \leq 3.556$
Symmetry	$x_4 = x_5 = -x_3 = -x_6$
	$y_4 = y_3 = -y_5 = -y_6$
	$z_4 = z_3 = z_5 = z_6$
	$x_8 = x_9 = -x_7 = -x_{10}$
	$y_8 = -y_9 = -y_{10}$

The optimization routines were executed for the seven approaches and the results obtained are summarised in table 6.9. The heaviest possible structure from assigning the biggest section resulted in a total weight of 510 kg.

Table 6.9: 25-Bar truss results

Approach	Time (s)	Result (kg)	Reduction (%)
Size	2.34	219.57	57.0
Topology	1.88	452.21	11.3
Shape	1.79	449.61	11.8
TS	2.01	220.50	56.8
STS	1.91	198.28	61.1
TSS	1.94	173.87	65.9
SIM	2.86	51.93	89.8

It is interesting to note that the size approach consumed more time than the other approaches, except for the SIM approach. The simultaneous approach again delivered the best result with a 89.8 % lighter solution than the original structure.

The performance of the approaches is shown in figures 6.6 and 6.7. By comparing figures 6.3 and 6.6 it can be seen that the performance of the optimization is fairly

## 6.2 25-Bar truss

similar. It is also worth noting the 4% difference between the results of the STS and TSS approaches. This indicates that their results are almost equivalent, with the main difference being the weight of the initial structure. The TSS initial structure has the same cross-section assigned to all the elements, while the STS initial structure's cross-sections are randomly initialized.

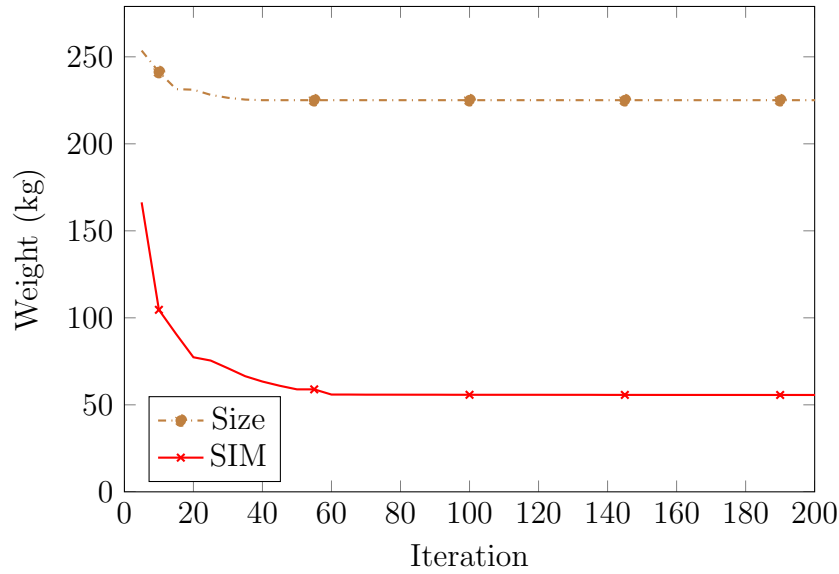


Figure 6.6: Performance of the size and SIM approaches for the 25-Bar truss

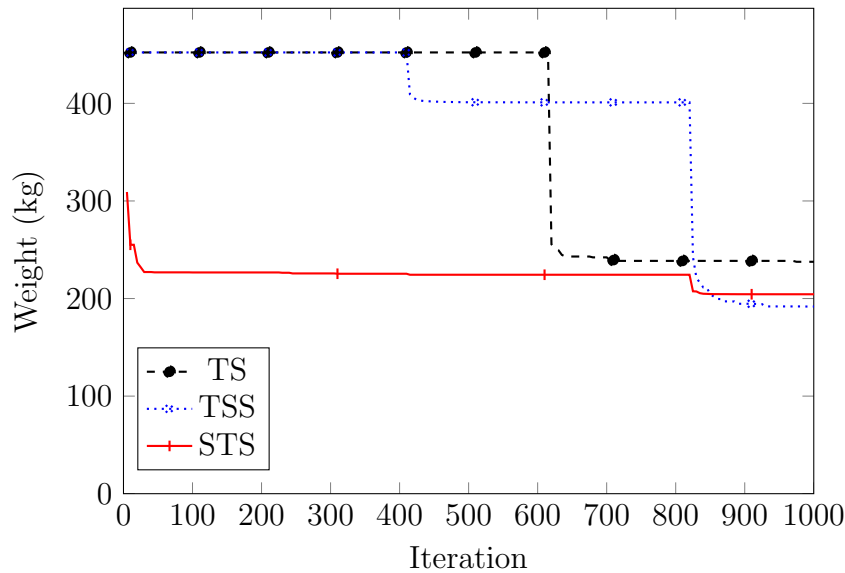


Figure 6.7: Performance of the TS, STS and TSS approaches for the 25-Bar truss

As a validity check of the results obtained, they are compared to the results presented in literature. For the size optimization approach, Toğan et al. (2008) and Coello et al. (1994) arrived at 219.3 kg and 224 kg respectively, which correlates well with the 219.6 kg found in this study. When considering the simultaneous approach, the 51.93 kg obtained is comparable to 50.7 kg found by Mortazavi et al. (2016).

### 6.3 47-Bar truss

The next benchmark structure is the two-dimensional 47-Bar truss shown in figure 6.8, with the element definitions given in table 6.10 and the list of the 50 available cross-sections shown in table B.3 in appendix B. This problem has been used by a number of researchers to test their developed algorithms (Mortazavi et al. 2016; Ahrari et al. 2015; Erbatur 2002).

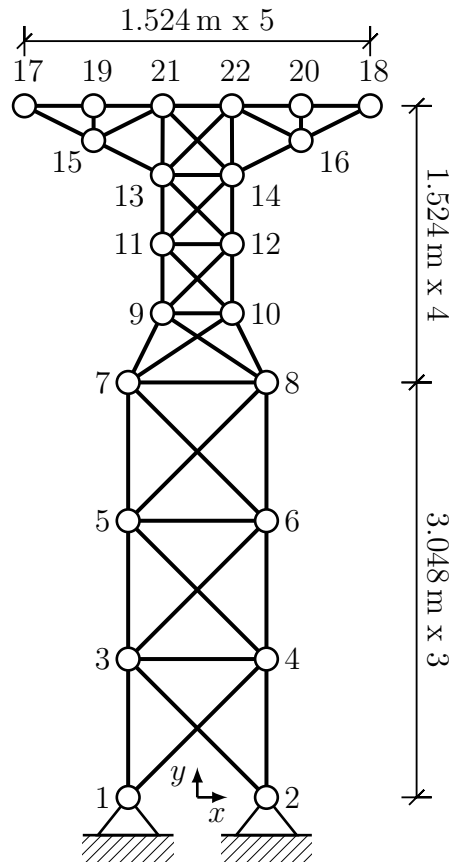


Figure 6.8: 47-Bar truss

Table 6.10: 47-Bar truss element definition

Element name (start node, end node)									
$A_1$	(1,3)	$A_{17}$	(9,11)	$A_{33}$	(15,21)	$A_2$	(2,4)	$A_{18}$	(10,12)
$A_{34}$	(16,22)	$A_3$	(2,3)	$A_{19}$	(10,11)	$A_{35}$	(17,19)	$A_4$	(1,4)
$A_{20}$	(9,12)	$A_{36}$	(18,20)	$A_5$	(3,5)	$A_{21}$	(11,13)	$A_{37}$	(15,17)
$A_6$	(4,6)	$A_{22}$	(12,14)	$A_{38}$	(16,18)	$A_7$	(4,5)	$A_{23}$	(12,13)
$A_{39}$	(14,21)	$A_8$	(3,6)	$A_{24}$	(11,14)	$A_{40}$	(13,22)	$A_9$	(5,7)
$A_{25}$	(13,21)	$A_{41}$	(21,22)	$A_{10}$	(6,8)	$A_{26}$	(14,22)	$A_{42}$	(13,14)
$A_{11}$	(6,7)	$A_{27}$	(13,15)	$A_{43}$	(11,12)	$A_{12}$	(5,8)	$A_{28}$	(14,16)
$A_{44}$	(9,10)	$A_{13}$	(7,9)	$A_{29}$	(19,21)	$A_{45}$	(7,8)	$A_{14}$	(8,10)
$A_{30}$	(20,22)	$A_{46}$	(5,6)	$A_{15}$	(7,10)	$A_{31}$	(15,19)	$A_{47}$	(3,4)
$A_{16}$	(8,9)	$A_{32}$	(16,20)						

What makes this problem interesting is that there is no displacement constraint. However, an additional buckling constraint, expressed in equation 6.2, along with differing allowable tensile and compression stresses are imposed on this problem. These constraints along with other design parameters are shown in table 6.11.

$$\sigma_{comp_i} \leq BEA_i/L_i^2$$

$$\text{with } i = 1, \dots, 47$$

$$B = 3.96$$
(6.2)

Table 6.11: 47-Bar truss design parameters

Parameter	Value
Young's modulus	206.84 GPa
Material density	8301 kg/m <sup>3</sup>
Allowable compressive stress	103.42 MPa
Allowable tensile stress	137.9 MPa

Another difference between the previous structures and the 47-Bar truss is that it is subjected to multiple load cases. These load cases are given in table 6.12. Intuitively more load cases lead to more analyses resulting in longer execution times. More load

**6.3 47-Bar truss**

cases also increase the complexity of the problem in terms of applying constraints and finding an optimal solution. Problems with multiple load cases are important to consider as it reflects a real-world scenario where structures are typically subjected to multiple load cases.

Table 6.12: 47-Bar truss loading conditions

Case	Nodes	$F_x$ (kN)	$F_y$ (kN)
1	17,18	26.69	-62.28
2	17	26.69	-62.28
3	18	26.69	-62.28

Symmetry about the y-axis is preserved by prescribing symmetrical nodes to have the same value while its counterpart is allowed to be a shape variable. These variables are shown in table 6.13. In total this problem consists of 27 size and topology variables and 17 shape variables which is significantly more than the previous two problems.

Table 6.13: 47-Bar truss variable detail

Variable	Detail
Size and topology	$A_m = A_{m-1}$
	with $m = 2, 4, 6, \dots, 40$
	$A_{41}, A_{42}, A_{43}, \dots, A_{47}$
Shape (mm)	$0 \leq x_2, x_4, x_6, x_8 \leq 3810$
	$0 \leq x_{10}, x_{12}, x_{14} \leq 2286$
	$0 \leq x_{20} \leq 3810$
	$0 \leq x_{22} \leq 2286$
	$0 \leq y_4 \leq 6096$
	$3048 \leq y_6 \leq 9144$
	$6096 \leq y_8 \leq 10668$
	$9144 \leq y_{10} \leq 12192$
	$10668 \leq y_{12} \leq 13716$
	$12192 \leq y_{14} \leq 15240$
	$13716 \leq y_{20}, y_{22} \leq 16764$
Symmetry	$x_2 = -x_1; x_4 = -x_3$
	$x_6 = -x_5; x_8 = -x_7$
	$x_{10} = -x_9; x_{12} = -x_{11}$
	$x_{14} = -x_{13}; x_{20} = -x_{19}$
	$x_{22} = -x_{21}$
	$y_4 = y_3; y_6 = y_5$
	$y_8 = y_7; y_{10} = y_9$
	$y_{12} = y_{11}; y_{14} = y_{13}$
	$y_{20} = y_{19}; y_{22} = y_{21}$

The results obtained from the various approaches are shown in table 6.14. The initial structure had a weight of 2989 kg and this was significantly reduced with the different optimization routines. The performance of the various optimization routines is shown in figures 6.9 and 6.10.

Table 6.14: 47-Bar truss results

Approach	Time (s)	Result (kg)	Reduction (%)
Size	12.41	1381.66	53.8
Topology	11.00	2683.97	10.2
Shape	16.68	2407.19	19.5
TS	12.16	1422.37	52.4
STS	13.26	1420.75	52.5
TSS	15.16	1322.23	55.8
SIM	18.94	909.48	68.6

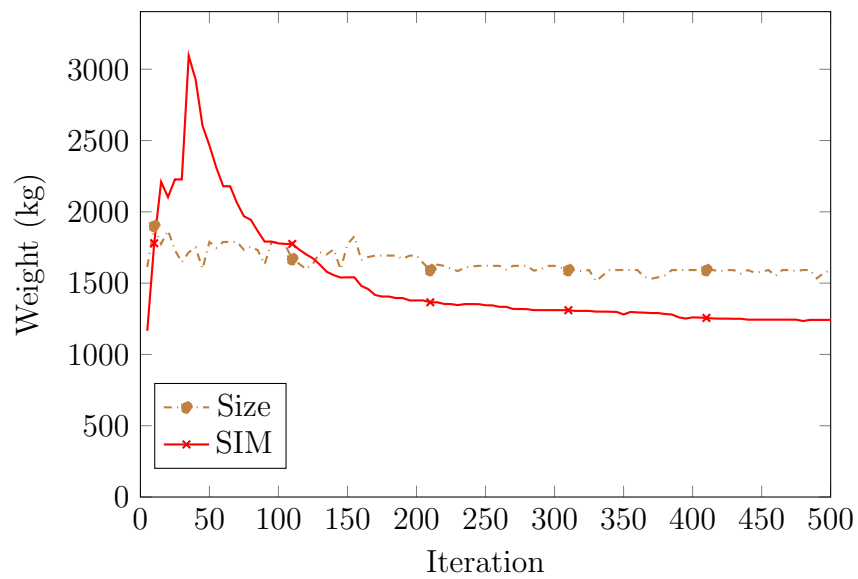


Figure 6.9: Performance of the size and SIM approaches for the 47-Bar truss

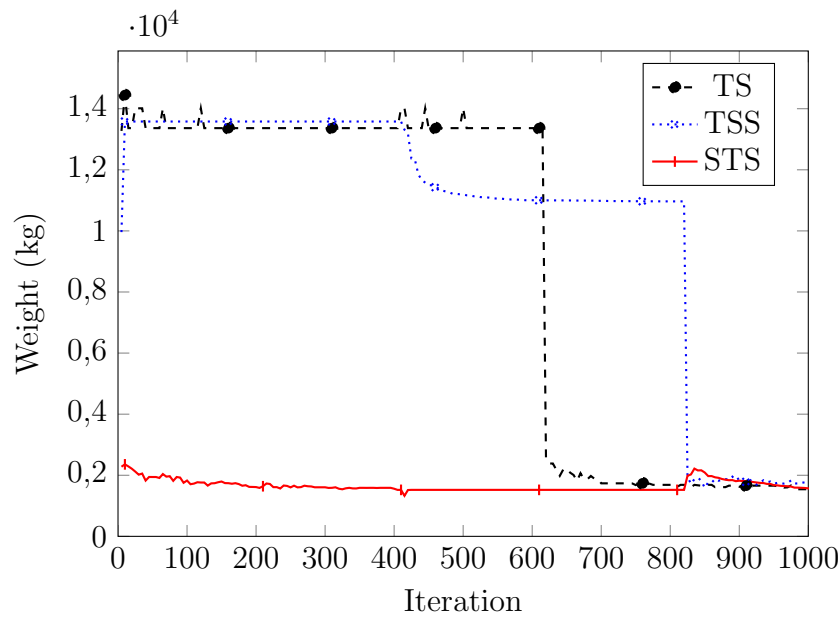


Figure 6.10: Performance of the TS, STS and TSS approaches for the 47-Bar truss

The resulting structure obtained from SIM optimization had a weight of 909 kg. This value is 8.7 % more than the 837 kg from Gholizadeh (2013) and 13.5 % more than the 801 kg reported by Mortazavi et al. (2016). The resulting weight difference between these papers may be attributed to the use of a better suited algorithm for a larger search space for the continuous shape variables compared to the GA used in this study.

Similar to the previous structures, the simultaneous optimization routine produced the lightest result. However, the SIM routine required significantly more time to arrive at a solution for the same number of iterations. This indicates that there is an additional computation involved when optimizing a structure simultaneously as opposed to a staged approach.

It is interesting to note that, for this problem, the size approach produced lighter solutions than the TS and STS approaches. This implies the inclusion of topology optimization, when run independently, results in a worse solution. It is also worth noting that this phenomenon only occurs for this example and it may be attributed to the nature of this specific problem. Another reason for this phenomenon may be due to the nature of the sequential approach. While the one stage improves the best structure, the next may negate the improvement. For example, the size stage may



select a good member, then during the next topology stage the member is removed from the structure.

## 6.4 72-Bar truss

The 72-Bar space truss, shown in figure 6.11, was optimized for size and topology by Kaveh (2013) by applying both static and dynamic constraints. In this case only static constraints are applied, but the shape of the structure is also optimized.

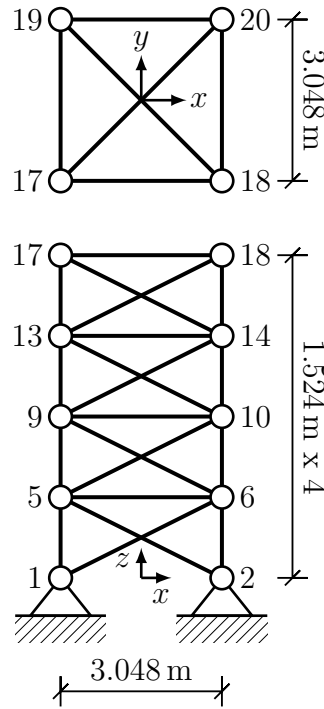


Figure 6.11: 72-Bar truss

The design parameters along with the displacement and stress constraints used in this problem are shown in table 6.15. The element grouping for the 16 size and topology variables is shown in table 6.16. The 64 cross-sections used for this problem was taken from Kaveh, Kalatjari, et al. (2016) and are shown in table B.4 in appendix B.

Table 6.15: 72-Bar truss design parameter

Parameter	Value
Young's modulus	68.95 GPa
Material density	2768 kg/m <sup>3</sup>
Allowable stress	172.38 MPa
Allowable displacement	6.35 mm

Table 6.16: 72-Bar truss grouping

Group	Element name (end nodes)
A1	1(1,5), 2(2,6), 3(3,7), 4(4,8)
A2	5(2,5), 6(1,6), 7(2,7), 8(3,6), 9(3,8), 10(4,7), 11(1,8), 12(4,5)
A3	13(5,6), 14(6,7), 15(7,8), 16(5,8)
A4	17(5,7), 18(6,8)
A5	19(5,9), 20(6,10), 21(7,11), 22(8,12)
A6	23(6,9), 24(5,10), 25(6,11), 26(7,10), 27(7,12), 28(8,11), 29(5,12), 30(8,9)
A7	31(9,10), 32(10,11), 33(11,12), 34(9,12)
A8	35(9,11), 36(10,12)
A9	37(9,13), 38(10,14), 39(11,15), 40(12,16)
A10	41(10,13), 42(9,14), 43(10,15), 44(11,14), 45(11,16), 46(12,15), 47(9,16), 48(12,13)
A11	49(13,14), 50(14,15), 51(15,16), 52(13,16)
A12	53(13,15), 54(14,16)
A13	55(13,17), 56(14,18), 57(15,19), 58(16,20)
A14	59(14,17), 60(13,18), 61(14,19), 62(15,18), 63(15,20), 64(16,19), 65(13,20), 66(16,17)
A15	67(17,18), 68(18,19), 69(19,20), 70(17,20)
A16	71(17,19), 72(18,20)

The structure is subjected to two load cases, each applying a different stress pattern within the structure. These load cases are specified in table 6.17.

Table 6.17: 72-Bar truss loading conditions

Case	Nodes	$F_x$ (kN)	$F_y$ (kN)	$F_z$ (kN)
1	17	22.25	22.25	-22.25
2	17,18,19,20	-	-	-22.25

With regards to shape optimization, the nodes on each level are allowed to vary between 0.5 m and 2.5 m in both the  $x$  and  $y$  directions, with no movement in the  $z$  direction. The other three nodes in the level are subsequently changed to maintain symmetry of the vertical structure. A total of 10 shape variables are, therefore, introduced to the problem.

The results from the various optimization routines are given in table 6.18. The base structure used has a weight of 626.9 kg. This is not the heaviest structure possible from the selection of sections, but given the large range of section sizes and the results obtained, a lighter structure which also satisfies the constraints was selected for the comparison.

Table 6.18: 72-Bar truss results

Approach	Time (s)	Result (kg)	Reduction (%)
Size	10.30	181.27	71.1
Topology	9.12	410.96	34.4
Shape	9.06	408.28	34.8
TS	9.83	243.6	61.1
STS	8.95	154.31	75.4
TSS	9.5	166.02	73.5
SIM	11.97	100.84	83.9

When comparing the result of 181 kg obtained for the size optimization with the 170 kg found by several other researchers (Jalili et al. 2015; Degertekin 2013; Camp 2007), there is a 6 % deficit. This may be due to a grouping discrepancy between the respective problem definitions and the use of a better suited algorithm with calibrated parameters. Unfortunately, no results to the SIM approach have been

## 6.4 72-Bar truss

published for the 72-Bar truss and the results obtained in this study can not be compared to ones from literature.

The performance of the individual routines is shown in figures 6.12 and 6.13.

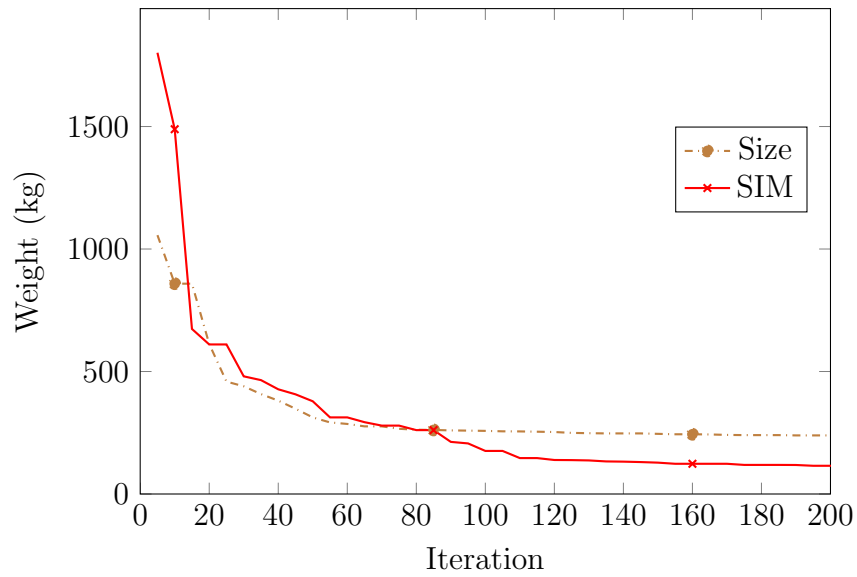


Figure 6.12: Performance of the size and SIM approaches for the 72-Bar truss

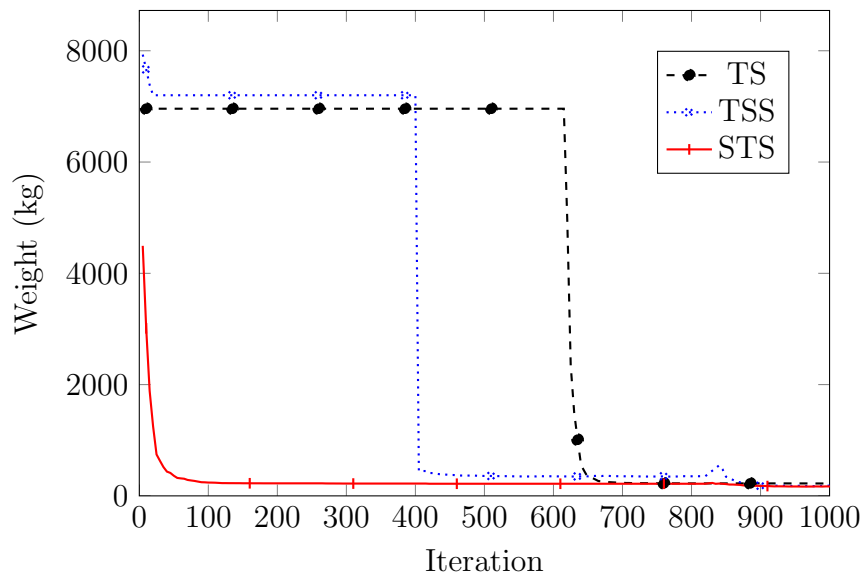


Figure 6.13: Performance of the TS, STS and TSS approaches for the 72-Bar truss

## 6.5 Combining results

From the results obtained, the well-known statement that considering the size, shape and topology aspects of a structure simultaneously produces the lightest structures, is validated. Through the quantification used in this chapter, it is concluded that the simultaneous approach yields, on average, a 13 % better solution than its best alternative, but requires additional computation time to complete.

Comparing only the individual approaches, size optimization clearly leads to better results, but consumes more time. From the results obtained in this study, the weight improvement is approximately 32 %. The reason for this can be attributed to the fact that the choice of cross-section has a significant influence on the weight of the structure, while removing certain non-critical elements and moving joints has a limited influence on the weight of the structure.

The staged approaches typically produce reasonable results with the same amount of iterations as the SIM approach. However, the iterations allowed for each stage is rather limited when each routine is forced to have the same total number of iterations. It is interesting to note that there is, on average, a 12 % difference between considering the three aspects in a staged manner as opposed to considering them simultaneously. The separation of the size, shape and topology aspects of the structure may be the cause for this difference since these aspects are not independent when it comes to the performance of the structure.

It is possible to quantify from the results obtained in this study that the simultaneous approach produces, on average, 22 % more economical structures than the size approach. It also always arrived at a better result than any of the considered staged approaches. This indicates that in search of a truly optimal structure, simply performing a size optimization is insufficient and that significant savings in terms of weight can be made by upgrading the optimization routine's complexity by considering more aspects of the structure. Based on the results in this study, it is evident that it is worthwhile to devote the additional effort to apply a more complex optimization approach in favour of a significantly better solution.

## 7. Multi-objective quantification study

The serviceability limit state (SLS) sometimes governs the design of a structure. The SLS is used to verify that the structure does not exceed the prescribed displacement limit. In the case where the deflection limit is exceeded, the structure is stiffened by increasing the size of its members, resulting in a heavier structure. However, in several instances the deflection of a structure can be regarded as a non-critical factor. In other words, a structure's deflection will not influence the usability thereof. For example, a rural warehouse structure will remain functional even if the deflection is occasionally greater than the prescribed limit.

It can be argued for a number of structures, such as the warehouse in the aforementioned example, the deflection limit may be increased in favour of a significant cost reduction. In this study, an attempt is made to quantify the amount of weight, if any, that can be saved by increasing the allowable deflection limit. It is assumed that the weight of a structure can be used as an indication of its cost.

The quantification is performed by introducing the two objectives, weight and deflection, to a multi-objective problem and minimizing them simultaneously. The resulting pareto front of solutions is used to determine whether or not increasing the deflection limit will lead to a meaningful weight reduction in the structure.

For this investigation, moment resisting frame structures are considered. The FEA and optimization module discussed in sections [5.1](#) and [5.2](#) are used for this purpose. To determine the capacity of a frame, a structural design module is introduced to automatically design the generated solutions. Further details of this module are presented in section [7.1](#).

With tools to determine the capacity of a frame structure, a formal definition of the multi-objective problem is required. The definition used in this study is given in section [7.2](#). It outlines the objectives and constraints present in this study and how

they are determined using the developed software.

Four structures are considered for this study. Their definition along with their respective results are discussed in section 7.3, after which a conclusion is drawn from these obtained results.

## 7.1 Automatic design module

Any steel structure is required to meet the requirements provided by the national standards of a country. In this study, SANS 10162-1 is used to determine if a structure can resist the applied loadings at the ultimate limit state (ULS). Considering the large number of structures that must be designed during an optimization, it is infeasible to design them by means of hand calculations. Therefore, the addition of a module which can automatically design structures is required.

SANS 10162-1 provides guidelines for determining the resistance of compression, tension and bending elements. Interaction equations and further considerations are also provided for an element subjected to a combination of forces. These guidelines are implemented to design all the elements in a structure.

With regards to the analysis of the structure, SANS 10162-1 stipulates that a second order analysis is required to determine the element forces. In this study, the analysis module, described in section 5.1, only caters for linear-elastic analyses. For the purposes of this study the use of a linear-elastic analysis is deemed sufficient, although analysis results may be improved by utilising a second order analysis.

A number of values used for the design of members, such as effective length factors, are difficult to determine from a software perspective. Therefore, these values should be explicitly defined for all elements before the design can be done. These values remain constant for all the solutions within the optimization routine and can simply be assigned for all of them. The values which must be defined before the optimization are listed below:

- Effective length factors for compression,  $K_x$ ,  $K_y$  and  $K_z$ .
- Member lengths,  $L_x$ ,  $L_y$  and  $L_z$ .

## 7.2 Formal problem definition

- Whether a bending member is laterally supported or unsupported. In the case the member is laterally unsupported, the effective length factor of the unsupported length,  $K_b$  is required.

One value that is particularly difficult to determine from the software's perspective is the  $\omega_2$  value which applies to laterally unsupported bending members. A formula for  $\omega_2$  is, however, presented in [CSA S16-09 \(2009\)](#), which can be determined by the software. This formula is given in equation [7.1](#) and is based on the quarter point moments within the element, namely,  $M_a$ ,  $M_b$  and  $M_c$ . This formula is accommodated by using the `MeshModel` described in section [5.1.4.3](#).

$$\omega_2 = \frac{4 \cdot M_{max}}{\sqrt{M_{max}^2 + 4 \cdot M_a^2 + 7 \cdot M_b^2 + 4 \cdot M_c^2}} \quad (7.1)$$

The results of the design module are validated using the provided reporting functionality. Using this function, a comprehensive design calculation sheet for the entire structure can be generated for checking the design. A small example of such a design sheet for the structure and loading shown in figure [7.1](#) is added to appendix [E](#). This is the same structure for which the analysis report, shown in appendix [C](#), is generated, hence the analysis and design reports are for the same structure.

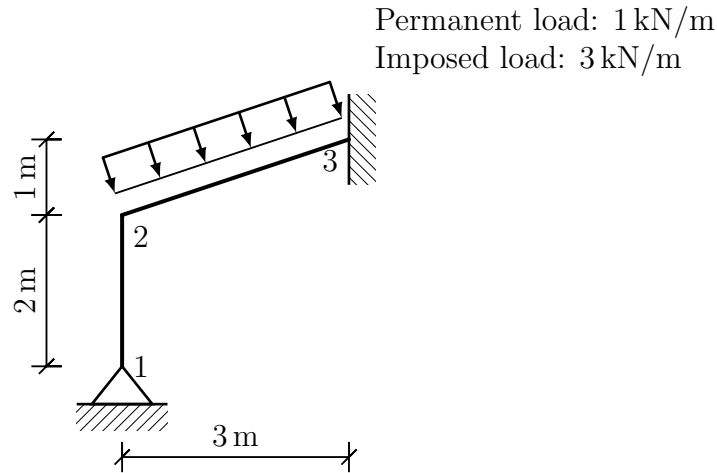


Figure 7.1: Example structure for the design report

## 7.2 Formal problem definition

The formal definition of the multi-objective problem is expressed in equation [7.2](#) and applied to all the problems considered in this study. In the aforementioned



## 7.2 Formal problem definition

expression, the two objective functions,  $W(\mathbf{x})$  and  $D(\mathbf{x})$ , correspond to the weight and displacement of the structure respectively. The constraint,  $C_1$ , is added to avoid allowing excessive deflection within the structure. This allows the optimization routine to be guided in a favourable direction. The second constraint,  $C_2$ , specifies that the structure must adhere to the requirements set by the design code. These requirements are focused on the load carrying capacity of the structure.

$$\begin{aligned} \text{minimize } W(\mathbf{x}) &= \sum_{i=1}^m \rho_i l_i A_i & (7.2) \\ D(\mathbf{x}) &= \max(\text{displacement}) \\ \text{subjected to:} \\ C_1 &\equiv \max(\text{displacement}) \leq 1 \text{ m} \\ C_2 &\equiv \text{Satisfies code requirements} \end{aligned}$$

For the problems considered in this study, only size optimization is used. This is done based on the assumption that the other aspects of the structure have been defined during other phases of its design. Therefore, only cross-section options can be defined for variable elements within these structures.

In this study, the optimal structure is defined as the one having the best trade-off or balance between the objectives. It is noteworthy that this is only one consideration as many others exist, for example simply considering the optimal as the lightest structure which satisfied the deflection and resistance requirements. In the case of two objectives, the result which best suits this balance definition is regarded as the point closest to a  $45^\circ$  tangent to the pareto front. This technique can be applied by normalising the weight and displacement axes to range from zero to one. This will accurately determine the solution of interest. This concept can be visually illustrated as in figure 7.2.

## 7.2 Formal problem definition

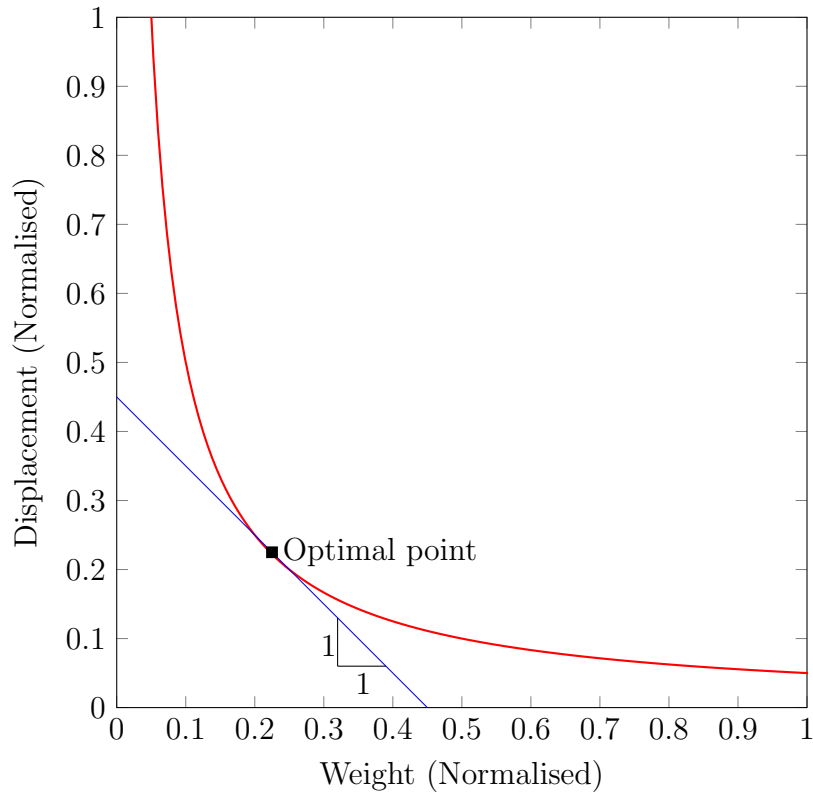


Figure 7.2: Definition of optimal solution

The merit of this definition of optimality is evaluated in the example structures by comparing how close the optimal structure is to the allowable deflection limit. In the case that the optimal solution exceeds the limit, it can be argued that the limit may be adjusted in favour of having an optimal structure. The opposite case of the optimal structure's deflection being less than the limit can be used as an indication that the optimal definition is not ideal and that a more suitable definition is required.

In all the examples of this study, a single, ULS, load combination is used to determine whether or not the structure has sufficient capacity to carry the applied load. For the allowable deflection calculation, another load combination, called SLS, is used to determine the maximum deflection within the structure. Therefore, each structure is analysed for two load combinations. In reality more combinations need to be considered, but one ULS and SLS combination is sufficient for the purpose of this study.

## 7.3 Example structures

A total of four structures are considered as examples of multi-objective optimization problems where both weight and displacement are minimized. These include two- and three-dimensional structures subjected to different loading and support conditions.

The parameters used for the NSGA-II optimization algorithm include a population size of 80 individuals and a maximum of 100 000 objective function evaluations which result in 1250 iterations throughout the optimization. The member grouping and available cross-sections for each variable is specified for each respective example problem.

The weight of the structure is determined in a manner similar to the one used in chapter 6. The density of steel is taken as  $7860 \text{ kg/m}^3$  and is multiplied by the volume of each element to produce the resulting weight of a candidate solution structure. Furthermore, S355JR steel is used with a yield stress,  $f_y$ , and Young's modulus,  $E$ , of 355 MPa and 200 GPa respectively.

With regards to the values required for the automatic design of elements, the unsupported lengths are taken as the distance between the element's end-nodes as it is assumed that sufficient lateral support is provided at these points. All the effective length factors are chosen as one for simplicity. This choice is considered to be conservative given no element in the example structures considered in this study has translation free ends, which would increase its effective length. In other words, all the effective length factors for the elements are actually less than one, decreasing its effective length which would result in a larger resistance value.

### 7.3.1 Plane 4-storey frame

The first example structure is a two-dimensional 4-storey frame. An illustration of the frame and its applied loads is shown in figure 7.3. This structure is based on the example used by Barraza et al. (2017) for minimizing the weight and inter-storey drift under seismic loads.

## 7.3 Example structures

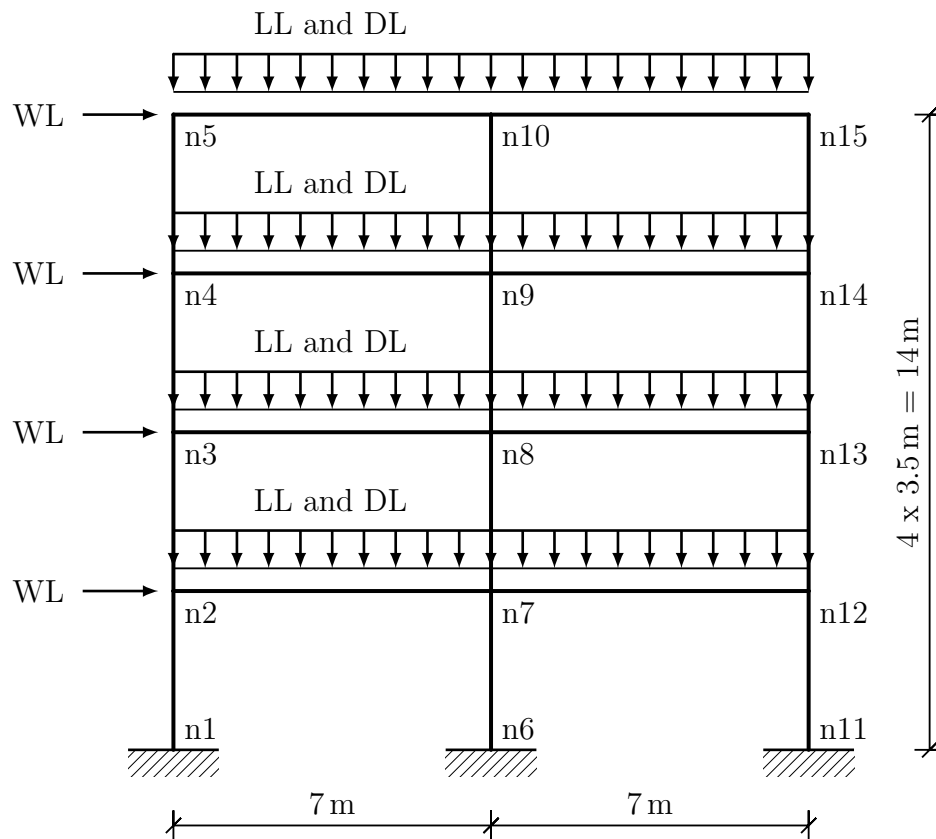


Figure 7.3: 4-Storey plane frame with loads

The loads applied to the structure include its own weight, DL, the imposed load, LL, and a wind load case, WL. The magnitudes of these loadings are shown in table 7.1. These loadings need to be combined to collectively produce the ULS and SLS load combinations. These combinations are expressed in equations 7.3 and 7.4.

Table 7.1: 4-Storey frame loading magnitudes

Load	Value
Own weight (DL)	3.0 kN/m
Imposed load (LL)	8.0 kN/m
Wind load (WL)	25.0 kN

$$ULS = 1.2 \cdot DL + 1.6 \cdot 0.3 \cdot LL + 1.3 \cdot WL \quad (7.3)$$

$$SLS = 1.1 \cdot DL + 1.0 \cdot 0.3 \cdot LL + 0.6 \cdot WL \quad (7.4)$$

### 7.3 Example structures

For this example structure, the weight and lateral displacement induced by the wind loads are minimized. The limit placed on the lateral deflection is specified by [SANS 10162-1](#) as height/400 which results in a limit of 35 mm.

With regards to grouping applied to the optimization problem, the two columns from  $n1$  to  $n5$  and  $n11$  to  $n15$  are grouped together and may comprise of either I- or H-sections. This leaves the elements in the middle column from  $n6$  to  $n10$  to be grouped together, with the same cross-section options. The beams of the structure are grouped such that two floors have the same section. In other words, the beams between nodes  $n2$  and  $n12$  on the first floor and  $n3$  and  $n13$  on the second floor are specified to have the same cross-section. This leaves the beams on the last two floors to be the last grouping for this structure. All the beams may consist of any I-section from the database.

The multi-objective optimization results obtained are shown in figure 7.4. The displacement limit of 35 mm is also indicated on the graph.

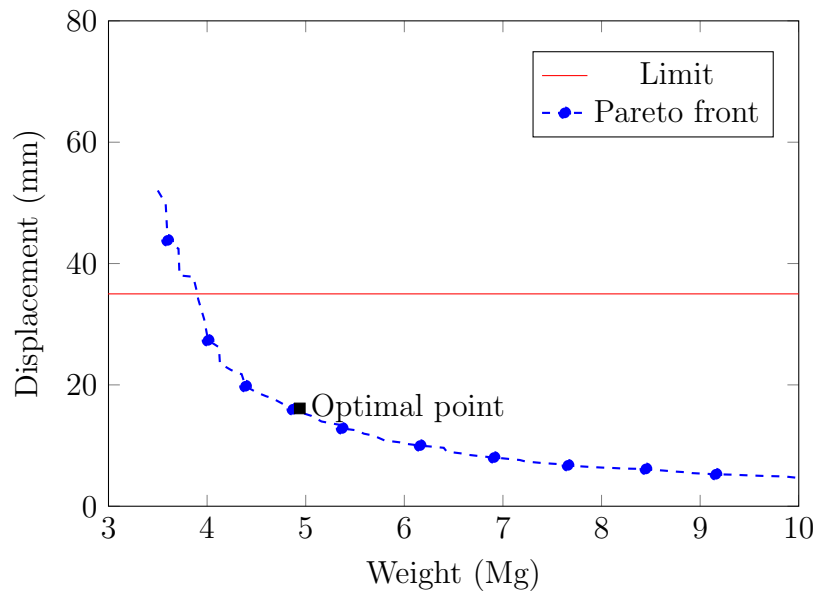


Figure 7.4: Resulting pareto front of the two-dimensional 4-storey frame

From the results, the solution which is the best trade-off between the two objectives has a weight of 4.66 Mg and a displacement of 17.82 mm. The found cross-sections for this optimal solution is shown in table [F.1](#). The displacement of the optimal solution

## 7.3 Example structures

---

is well below the limit of 35 mm. However, the structure which is the closest to the limit has a deflection of 34.1 mm and a corresponding weight of 3.91 Mg, leading towards a 16 % reduction of 750 kg from the predefined optimal solution.

For the case where the prescribed limit can be altered the weight can be reduced between 20 kg and 410 kg with the maximum displacement ranging between 35.8 mm and 52.04 mm. These ranges indicate that a meaningful weight reduction of up to 10 % can be achieved by allowing the deflection to exceed the prescribed limit. However, for this reduction the limit must be approximately doubled.

It is important to note that the change in structural weight is not taken into account during the optimization. In other words, the value of the permanent load, DL, applied to the structure is not recalculated for every new solution. If the resulting structure is re-analysed with its actual own weight forming part of the permanent load, the deflection will be reduced.

### 7.3.2 Plane portal frame

The next example is a typical portal frame structure. A portal frame is a popular structure normally used for industrial purposes such as warehouses and factories. The layout and applied loads of the two-dimensional portal frame in this example is shown in figure 7.5.

Portal frame structures normally function by allowing individual frames to carry their in-plane loads and installing a bracing system for dealing with lateral loads. It is efficient to determine which frame is the most heavily loaded and only designing that specific frame in detail and using one frame throughout the structure. Therefore, only the single most heavily loaded individual frame within such a portal frame structure needs to be optimized.

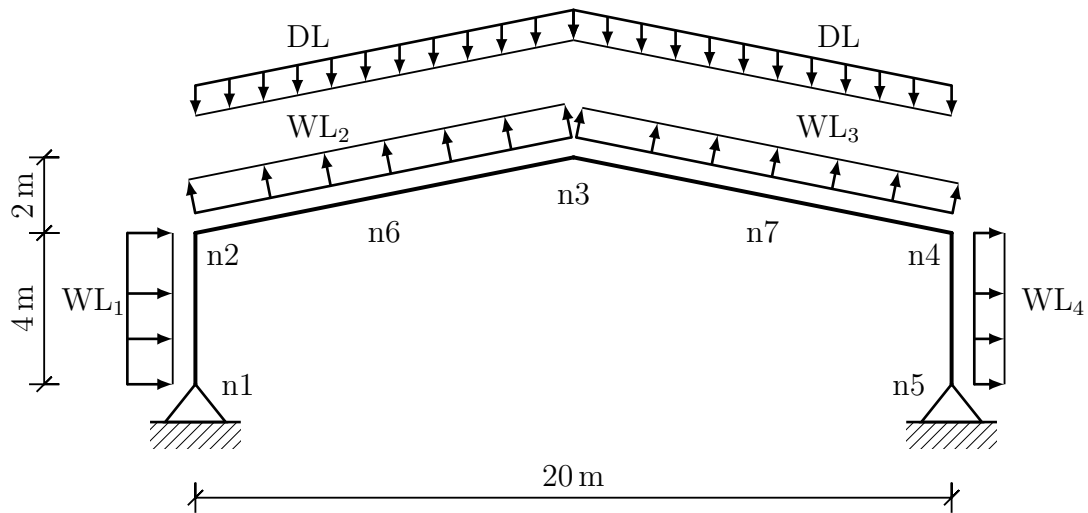


Figure 7.5: Plane portal frame with loads

For the frame shown in figure 7.5, it is assumed that adequate lateral support is provided to both flanges at all nodal positions. The nodes n6 and n7 are both in the middle of the rafter beams. The positioning of lateral support is specifically important for the structural design of elements for both bending and compression. The sections of an element between lateral supports are checked according to [SANS 10162-1](#) for flexural and torsional buckling.

The numerical values used for the indicated loadings are shown in table 7.2. For this structure it is assumed that the imposed load on the roof is negligible and that the permanent load, DL, includes both the own weight of the rafters and the roof which is resting on top of it. The two load combinations used during the optimization to determine the structure's capacity and deflection are shown in equations 7.5 and 7.6.

Table 7.2: Plane portal frame loading magnitudes

Load	Value
Permanent load (DL)	1.6 kN/m
Wind load (WL <sub>1</sub> )	5.9 kN/m
Wind load (WL <sub>2</sub> )	2.8 kN/m
Wind load (WL <sub>3</sub> )	3.8 kN/m
Wind load (WL <sub>4</sub> )	1.4 kN/m

### 7.3 Example structures

$$\text{ULS} = 1.2 \cdot DL + 1.3 \cdot WL \quad (7.5)$$

$$\text{SLS} = 1.1 \cdot DL + 0.6 \cdot WL \quad (7.6)$$

The grouping applied for the optimization of the portal frame is quite elementary. The two side columns are to have the same cross-section while the two rafter beams are prescribed to consist of the same cross-section. This yields an optimization problem with only two variable elements. All the I- and H-shaped sections are made available for the two columns, while the rafters may only consist of I-sections. All these available sections result in 2666 possible combinations for this portal frame.

The resulting pareto front from the optimization routine is shown in figure 7.6. According to [SANS 10162-1](#), the lateral deflection limit of an industrial portal frame can range between height/400 to height/200 which amounts to an allowable deflection range of 10 mm to 20 mm. In this instance the lowest deflection of 10 mm is used, which is the same as for non-industrial buildings. This allowable lateral deflection value is also indicated on the graph.

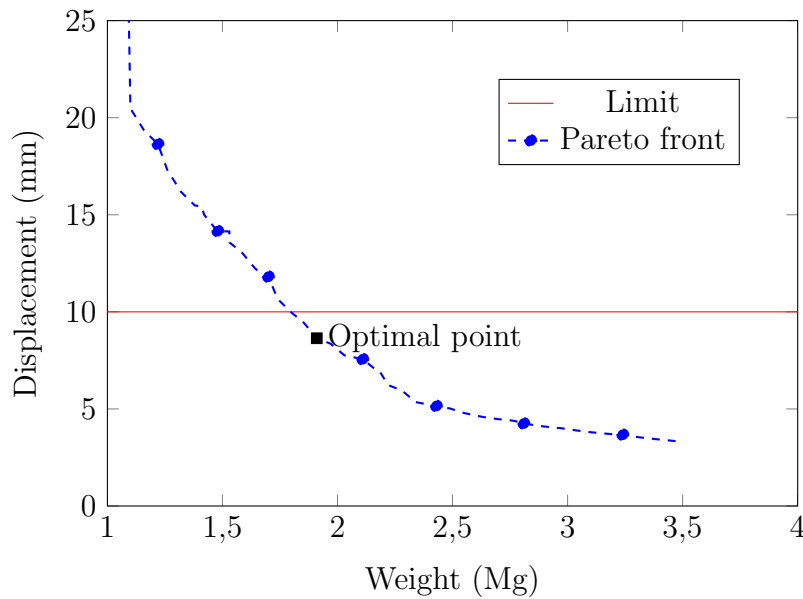


Figure 7.6: Resulting pareto front of the two-dimensional portal frame

By applying the previous definition of the optimal trade-off between the two objectives, the so-called “optimal” solution, identified from the pareto front, has a



## 7.3 Example structures

weight of 1.91 Mg and a lateral deflection of 8.6 mm. The location of this solution's data point on the resulting pareto front is marked on figure 7.6 and the corresponding cross-sections are shown in table F.2.

This optimal solution is well within the capacity requirements of SANS 10162-1, with the most critical element's interaction equation having a value of 0.29, indicating the structure has reserve capacity. However, its lateral displacement is very close to the 10 mm limit which indicates that the definition of the optimal solution is suitable for this example structure. The solution closest to the limit has a weight of 1.8 Mg and a lateral displacement of 9.96 mm. By comparing the optimal point with the solution closest to the limit, a 6 % weight reduction can be achieved. Keeping in mind that the deflection is only increased by a mere 1.4 mm.

If the limit is allowed to be increased to 15 mm, the weight of the structure decreases to 1.47 Mg, about 330 kg lighter than the solution which satisfied the limit. This amounts to a 18 % reduction in structural weight which can be regarded as a significant reduction.

If the higher limit of 20 mm was selected, then 76 of the 80 solutions obtained by the optimization are satisfactory. In such a case, simply the lightest solution of the population, which satisfies the constraints, could be used without any further consideration.

### 7.3.3 5-Bay portal frame

The first three-dimensional structure presented, in this chapter, is a typical portal frame structure. These structures are widely used to function as warehouses, retail facilities and agricultural buildings. The majority of the loadings on the structure are carried by the plane frames as in the previous example, while the out of plane loads, normally wind, are transferred through the structure to a bracing system.

The structure considered in this example has the same portal frame as in the previous example, spaced at 7 m intervals and connected via angle-sections and a cross-bracing system. This configuration is illustrated in figure 7.7.

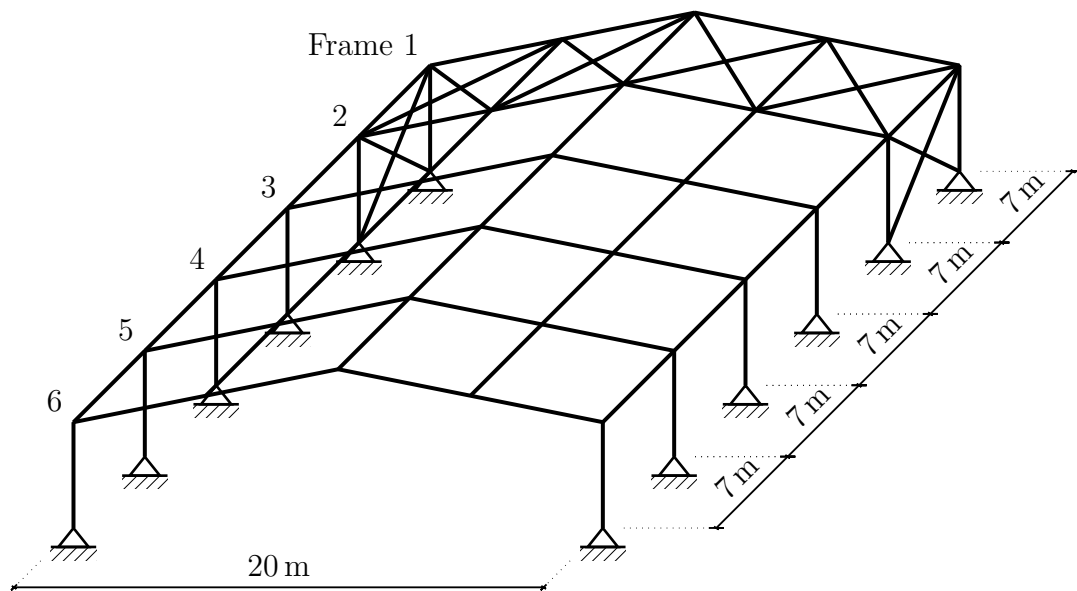


Figure 7.7: Three-dimensional portal frame

For the analysis of the structure, part of the bracing system is removed. This is done to ensure that the bracing elements only carry tensile forces in the considered load combination. If the transverse wind was acting in the opposite direction, the removed members will be required as they would then act in tension.

By ensuring that only tensile forces are carried by the bracing members, the limit placed on their slenderness ratios is increased from 200 to 300. This increase along with the members being braced at midspan, reduces the probability that their lengths are a limiting factor. If compressive forces are to be accommodated, these members would need to be shortened or lateral supports installed to avoid buckling. In a real structure these members will be present and compressive forces may be implicitly induced in these members. Although this may be the case, these elements will not be able to carry the compressive force and will buckle, leaving the force to be transmitted to the other bracing members to carry it in tension.

For the automatic design of this three dimensional structure, the effective lengths of the elements connecting the frames, with regards to compression, are assumed to be 3.5 m. This decision is justified by considering that these elements may be laterally supported by the roof sheeting or bracing elements added between these members to prevent buckling.

### 7.3 Example structures

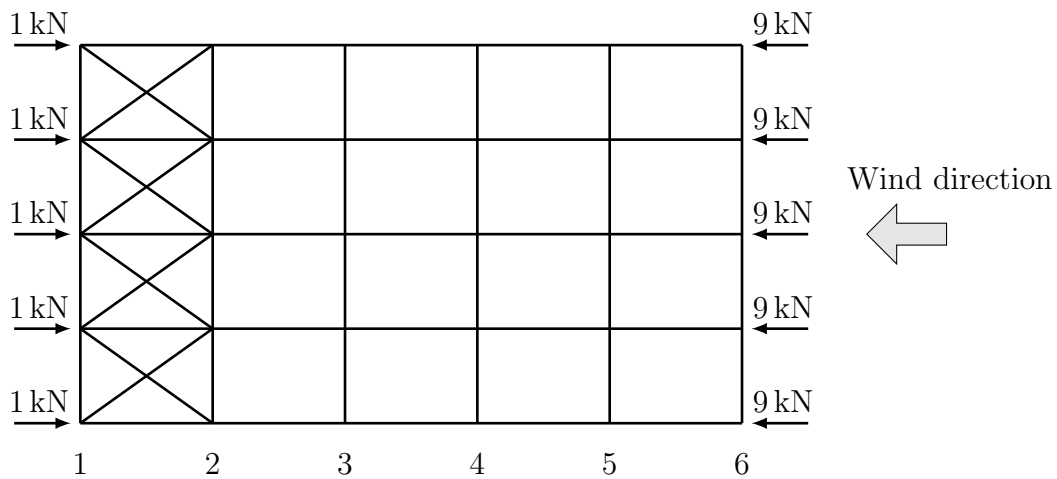
---

The wind load case applied to this structure is taken as the wind blowing on the face of the structure to induce forces in the bracing members as shown in figure 7.8a. By doing so, the maximum horizontal displacement is determined by the stiffness of the bracing elements and not the in-plane stiffness of the portal frames as in the previous example. For this wind load case, only the two faces perpendicular to the wind experience a positive pressure, while the rest of the structure is subjected to negative pressure. This causes a reduction in downward force in the structure which in turn increases the maximum lateral deflection.

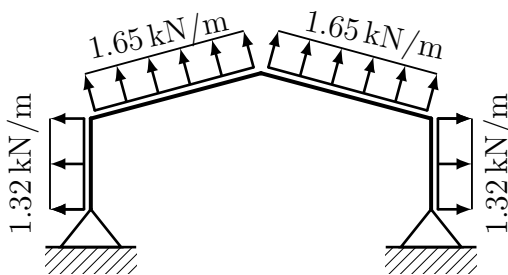
The loads induced by the applied wind case are shown in figure 7.8. Figure 7.8a shows the effect on the two windward faces. It is assumed that all the wind on these faces are transferred by the members connecting the frames and not the frames themselves. By doing so, the frames do not resist forces which causes bending about its weak axis for which a stiffer choice of cross-section would be required. The pressure on these faces are therefore converted to point loads of equal magnitude and applied at the nodes of the connecting members.

Figures 7.8a to 7.8e illustrate the forces applied to the various portal frames in this structure. These line loads were calculated by considering the variation of wind pressure on the roof of the building and that the surface area carried by each frame is equal to half the distance to the next frame on each side. Therefore, frames 2 to 4 are subjected to the same loads, while due to the increased pressure carried by frames 5 and 6 their roof loads would be higher. The two edge frames carry only half of the surface area compared to the other frames which in turn results in them carrying less load.

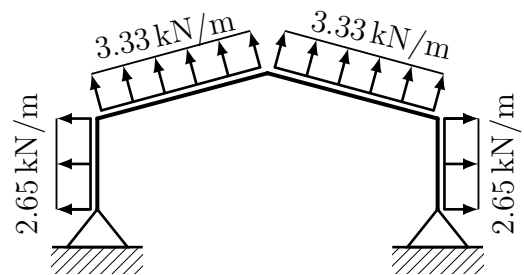
## 7.3 Example structures



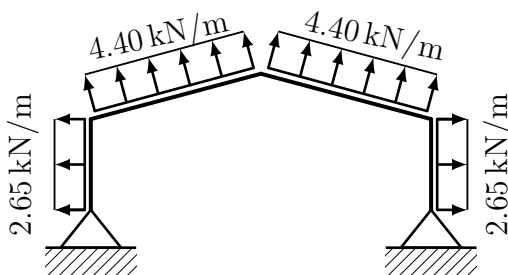
(a) Plan view with frame numbering and point loads



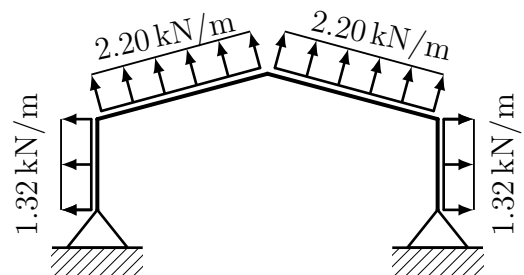
(b) Frame 1



(c) Frame 2, 3 and 4



(d) Frame 5



(e) Frame 6

Figure 7.8: The wind direction and induced loads on the three-dimensional portal frame

The same load combinations as the two-dimensional case were applied for the ULS and SLS cases. The value of the dead load applied to the rafters was slightly increased to 1.8 kN/m to account for the additional bracing elements. Doing so is not technically correct considering a number of members are now carrying more load than what is present because the braced bay is only situated in one bay. This approach is, however, considered to be conservative.

### 7.3 Example structures

With regards to the grouping applied to the structure, a total of eight groupings were created and is shown in figure 7.9. Six of which apply to the frames. The rafter beams and columns are paired from the outside inward, which also promotes symmetry in the structure. Each column grouping can use any I- or H-shaped cross-section, which amounts to 62 cross-sections, while the rafters may only comprise of one of the 43 I-sections. Furthermore, the elements connecting the frames are all grouped together while the final grouping is all the bracing members. Both these groupings may use any of the 46 equal-leg angle-sections during the optimization. All these groupings result in a total of  $40 \cdot 10^{12}$  possible solutions to the optimization problem.

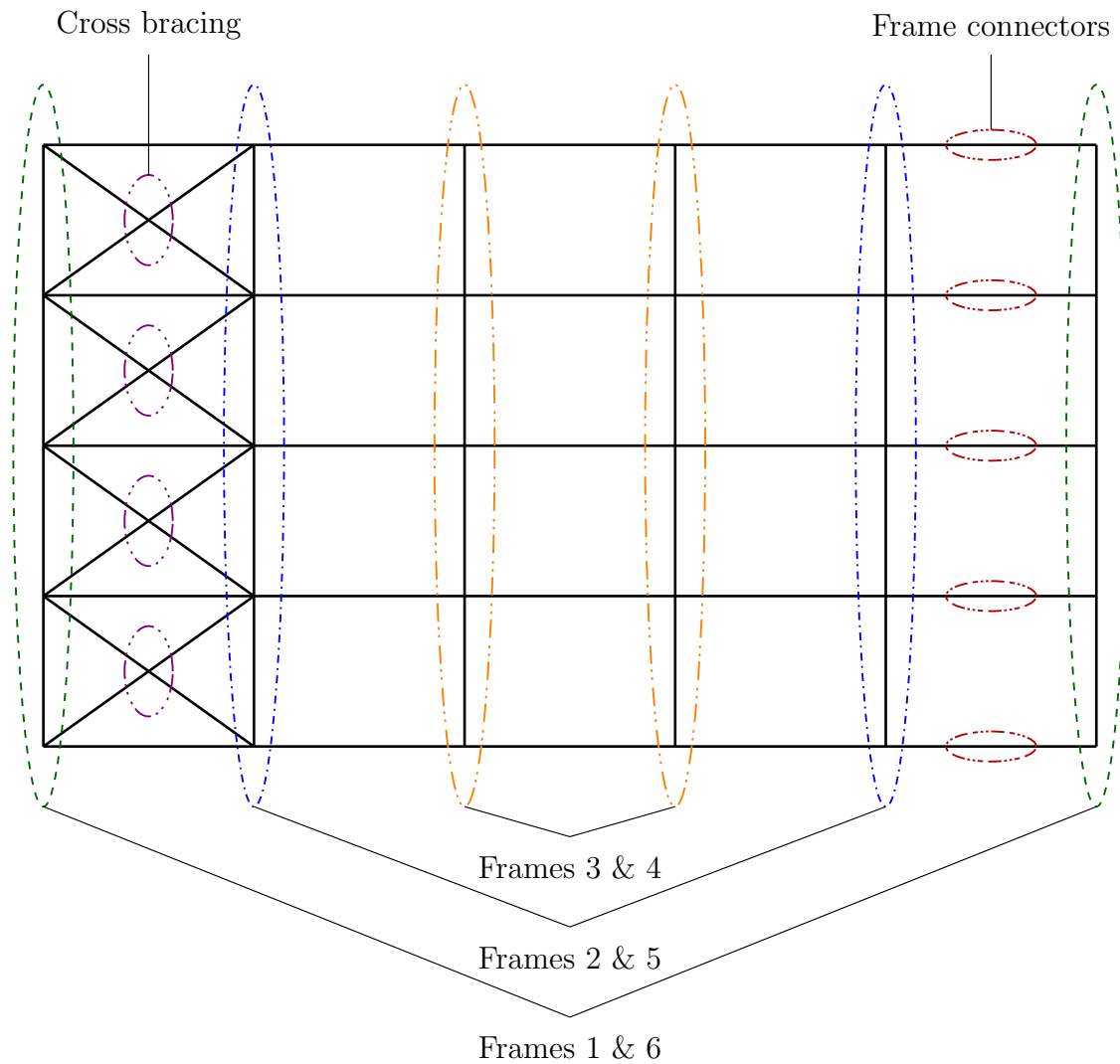


Figure 7.9: Grouping configuration applied to the three-dimensional portal frame

### 7.3 Example structures

The pareto front obtained from the optimization is presented in figure 7.10. As in the previous example, the deflection limit and the optimal structure as per the previous definition is indicated on the graph.

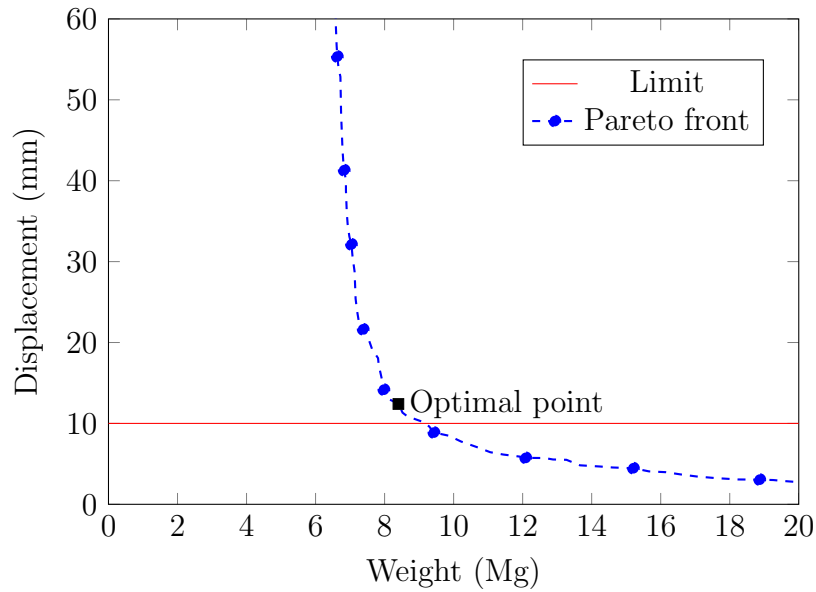


Figure 7.10: Resulting pareto front of the three-dimensional portal frame

The same horizontal displacement limit of 10 mm from the previous example applies to this structure. In this case, the maximum lateral displacement occurs in the windward direction, which is primarily resisted by the bracing members within the structure.

The lightest structure which conforms to the displacement limit has a weight of 9.29 Mg and a displacement of 9.6 mm. The optimal solution as previously defined slightly exceeds the displacement limit with a lateral displacement of 12.4 mm and a weight 8.4 Mg. The optimal solution is an 11 % reduction from the first structure which satisfies the displacement limit. The obtained cross-sections from this optimal solutions is shown in table F.3. For the case where the same adjusted limit of 15 mm, as suggested in the two-dimensional case, is applied the weight of the structure reduces to 7.98 Mg which amounts to a significant 16 % reduction.

It is also interesting to note that 37 of the 80 solutions on the pareto front exceed the displacement limit of 10 mm. Considering all the structures obtained in this pareto

## 7.3 Example structures

front satisfies the design requirements, the variability of the resulting weight can be vast. In this case, the displacement limit reduces the number of viable options which indicates that the prescribed displacement can be considered a limiting factor when searching for the optimal structure.

### 7.3.4 4-Storey building

The fourth example structure considered in this study is a rectangular four storey building which may be used as an office or residential building. This structure is considered as a three-dimensional finite element model with wind blowing on the structure. A diagram of this structure is shown in figure 7.11. This structure consists of four storeys, each with a height of 3 m, with each floor having an area of 288 m<sup>2</sup> which totals an area of 1440 m<sup>2</sup>.

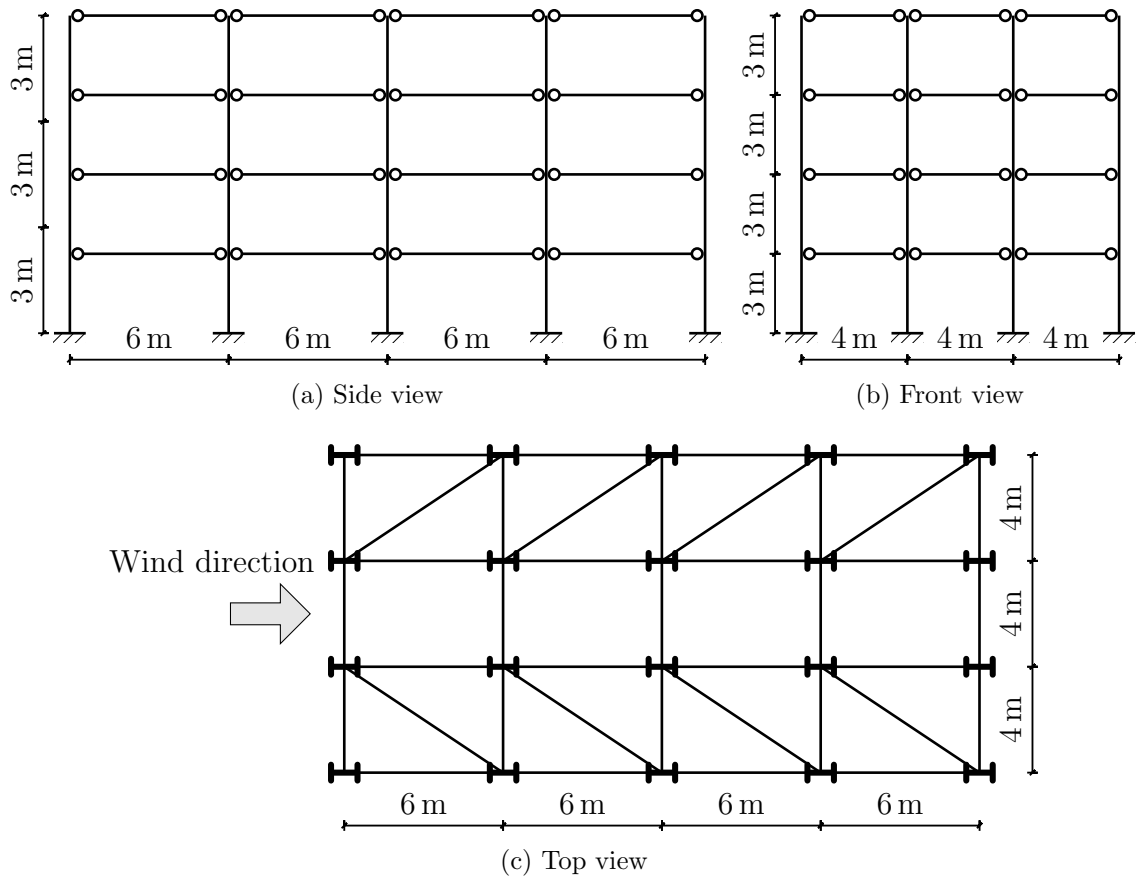


Figure 7.11: 4-Storey frame layouts

With regards to the structural considerations, all the columns are rigidly connected to

### 7.3 Example structures

---

the foundations which enables moments to be resisted at the supports. All the beams connecting to the columns are pin connected which ensures that no bending-moments can be transferred from the beams to their respective supporting columns. For this pin end-condition, however, torsional restraint must be provided in order to avoid a zero-pivot error during the analysis of the structure. This error arises when a beam with torsional stiffness is left unrestrained against torsion, which leads to the global matrix equation, equations 5.15 and 5.16, to be unsolvable. For this end-condition to be enabled, only the x- and y-axis moment degrees of freedom are released from the element stiffness matrix, leaving the torsional degree of freedom unaltered.

Bracing members are also present in the structure to provide lateral stability and reduce the maximum deflection. Since it is assumed that bracing only acts in tension, the bracing members which will be in compression during the considered load combinations were consequently removed from the analysis. This operation is needed to prevent slender bracing members in compression from entering the design stage, which they will fail due to buckling.

The orientation of the columns is also selected to improve lateral stiffness in the direction where bracing is not installed. By doing so the structure itself has sufficient lateral stiffness in one direction, while the bracing increases the overall stiffness in the other. In total the structure consists of 236 elements, which are all meshed into four sub-elements during the analysis in order to successfully execute the automatic design module.

In this example, the structure is subjected to three main loading scenarios. The first two include the permanent, DL, and imposed, LL loads. The third is a wind load, WL, blowing across the structure as indicated in figure 7.11c.

The wind load induces a load of 1.6 kN/m per floor on both the windward and leeward faces of the structure. This line load is converted to two point loads of each 9.6 kN on the outer beams of each floor as it is assumed that the wind load is transferred to the floor bracing which distributes it to the supporting columns on that floor, rather than allowing the load to be resisted by only the columns directly facing the wind which induces large deflections.



### 7.3 Example structures

The permanent load is applied to only the beams and it is assumed that the weight of the columns, bracing and floors are included in this loading. The imposed load includes the loads applied on the floors which are supported by the beams and the services installed in the structure such as air conditioning. The same beams which carry the permanent load is also required to carry an imposed loading.

A distinction is made between perimeter and internal beams since each will be required to carry a different part of the load transferring from the floor. Internal beams support a floor on both sides while perimeter beams only support one part of the floor. By utilising this assumption, the loadings applied to the beams are illustrated in table 7.3. One remark that should be made is that the top floor is also loaded similarly to the other floors and that the roof of the building is excluded from the model.

Table 7.3: Three-dimensional frame loading magnitudes

Load	Internal	Perimeter
Permanent load (DL)	17 kN/m	13 kN/m
Imposed load (LL)	14 kN/m	7 kN/m

With regards to the grouping applied to the structure as a multi-objective optimization problem, a total of six groups were identified to group elements of similar length and loading together. The applied grouping is the same for all levels and can therefore be represented in a simple manner as in figure 7.12. The various groups and their available cross-sections are shown in table 7.4.

Table 7.4: Grouping and section assignments for the three-dimensional 4-storey frame

Grouping	Assigned cross-section
Perimeter / outer columns	All I- and H-sections
Internal columns	All I- and H-sections
Perimeter/ outer beams	All I-sections
Lateral internal beams	All I-sections
Transverse internal beams	All I-sections
Bracing members	All Equal-leg angle-sections

### 7.3 Example structures

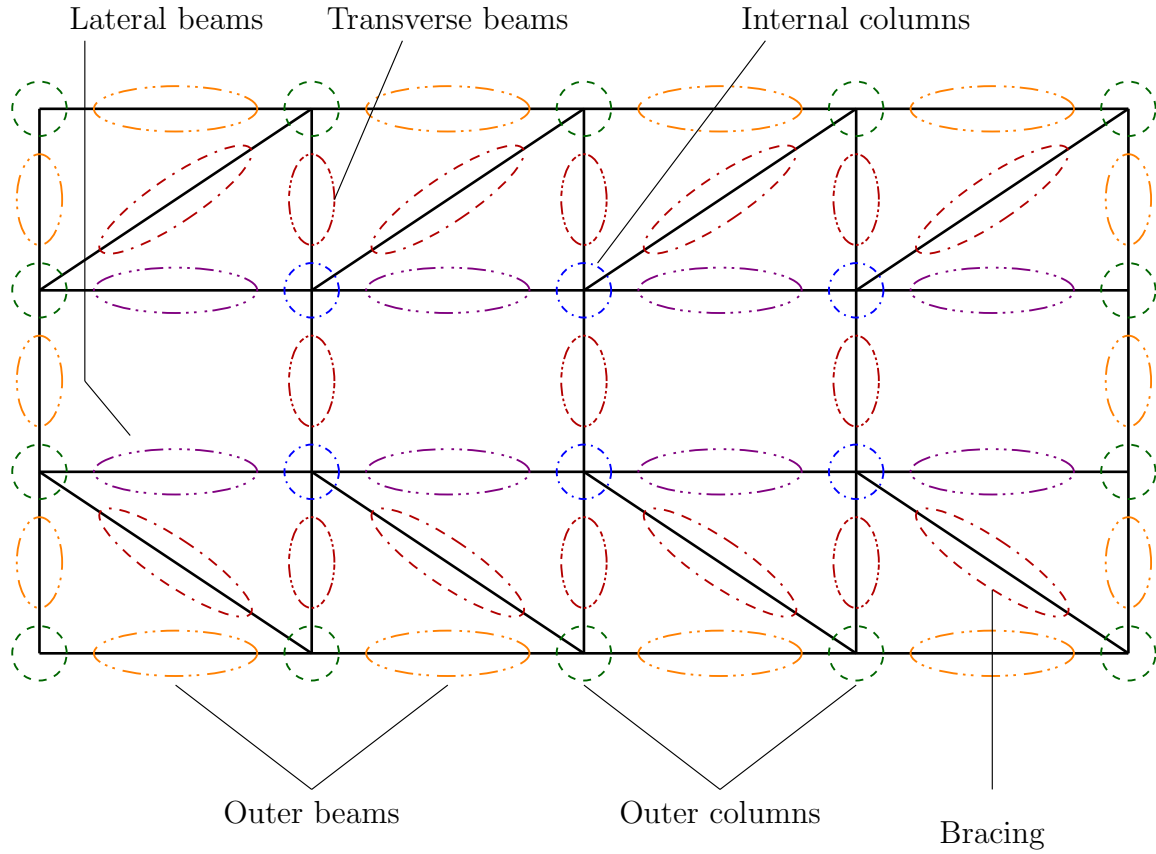


Figure 7.12: Grouping configuration applied to the three-dimensional frame structure

The load combinations applied to this structure are shown in equations 7.7 and 7.8. For ULS, the case where no wind is blowing on the structure is selected as this case induces larger bending moments in the beams. Although the columns are subjected to bending moments if the wind blows, the factor applied to the imposed load,  $LL$ , reduces to 0.48 which dramatically reduces the applied beam loadings, leading to a reduction in the design forces. Since lateral deflection is minimized for the SLS, it is appropriate to apply the maximum wind load with a reduced imposed load. This SLS combination promotes lateral deflection with a high lateral load and reduced downward force acting on the structure.

$$ULS = 1.2 \cdot DL + 1.6 \cdot LL + 0 \cdot 1.3 \cdot WL \quad (7.7)$$

$$SLS = 1.1 \cdot DL + 0.3 \cdot 1 \cdot LL + 0.6 \cdot WL \quad (7.8)$$

The resulting pareto front obtained from the optimization is presented in figure 7.13. The deflection limit of this structure is calculated as height/400 which yields a limit

### 7.3 Example structures

of 30 mm and the optimal structure as per the previous definition is indicated on the graph.

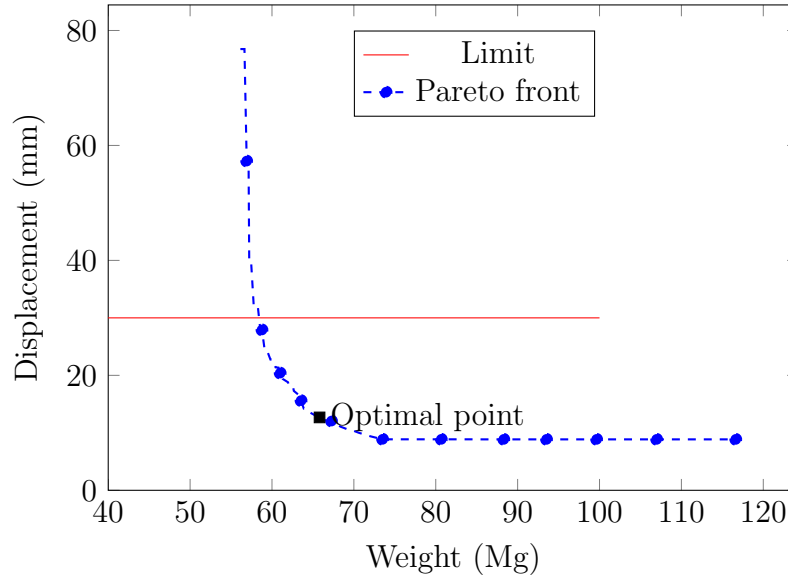


Figure 7.13: Resulting Pareto front of the 4-storey frame

The structure closest to the prescribed limit has a weight of 58.52 Mg and a displacement of 28.7 mm. It is notable that only eight solutions on this front exceed the limit with the maximum displacement reaching 76.8 mm while the remaining 72 solutions are well within the limit.

The optimal point as per the previous definition has a weight of 65.81 Mg and a displacement of 12.7 mm. The corresponding cross-sections obtained for this optimal solution are shown in table F.4. This solution is an 11 % weight increase from the one closest to the displacement limit. Therefore, the structure closest to the limit can be considered to be a significant weight reduction compared to the structure which results from the best trade-off optimality definition.

From the results obtained, the lightest solution with a 76.8 mm displacement has a weight of 56.1 Mg, a mere 4.1 % reduction in weight while the displacement is increased by 268 %. These numbers indicate that it would be infeasible to increase the displacement limit for such a small reduction of structural weight.

### 7.3.5 Concluding remarks

A total of four framed structures were considered as example problems where the weight and displacement are minimized by using multi-objective optimization. These structures were also required to satisfy the design requirements from [SANS 10162-1](#).

The aim of the investigation is to determine whether or not a significant weight reduction can be made by increasing the displacement limit set by [SANS 10162-1](#) for the serviceability limit state (SLS). The example structures show that a weight reduction of up to 16 % can be achieved when the displacement limit is increased. However, for the reduction to be significant the limit should be increased by a large margin, in a number of cases almost doubled, which makes the limit alteration infeasible.

The optimum solution is defined as the best trade-off between the two objectives and proves to be a reasonable solution in the considered examples. In a number of cases such as the two portal frame examples, the optimum solution's displacement is very close to the limit which supports the applicability of this definition of optimality. This definition of optimality may be used as an initial choice of the optimal structure and may be changed based on the requirements of the structure, for example, when a certain type of cross-section is preferred.

As a final remark, the displacement limit imposed on the structures considered in this study is not the governing factor when it comes to reducing the weight of the structure. In all the considered cases, the limit only removes a number of the lightest solutions on the pareto front from being considered with the weight difference being less than 20 %. This indicates that the majority of the structure's weight can be saved before reaching the prescribed limit and that the limit does not need adjustment for a significant weight reduction.

## 8. Conclusions and recommendations

### 8.1 Research overview

Research into the field of optimization has received attention over the years. Various methods for solving optimization problems have been developed including mathematical and non-mathematical techniques. These problems are typically solved with the aid of a computer due to their complexity and large number of iterations.

Research has been done to apply these methods for the optimization of structures. Generally, three aspects of a structure can be optimized, namely its size, shape and topology. Size refers to the choice of element cross-sections, shape to the nodal or boundary positioning and topology to the internal element configuration. A combination of these aspects can also be considered, be it sequentially or simultaneously.

Typically, only the size aspect of a structure is optimized, while a more complex approach can be followed such as considering the size, shape and topology aspects simultaneously. Although the majority of studies only optimize the size aspect of a structure, more complex approaches exist where the size, shape and topology of the structure is optimized simultaneously. This simultaneous approach is known to yield a more economical result. However, it is not known how much is gained by applying a more complex approach in terms of structural weight reduction. The first aim of this research is to investigate this improvement by means of a quantitative study.

Furthermore, multi-objective optimization enables the minimization of two or more conflicting objectives in an optimization problem, for example, the maximum displacement and weight of a structure. In this study, the second aim is to investigate whether or not the deflection limit placed on a structure by design codes should be increased in favour of a significant weight reduction. It is assumed that the weight of a structure can be used as an indication of its cost.

## 8.2 Consideration of objectives

For the successful execution of this study, the objectives identified in chapter 1 have been addressed as outlined below.

1. The first was to gain insight in the field of single- and multi-objective optimization in general as well as how standard optimization methods are adapted to suit structural optimization problems. Chapter 2 defines optimization in general and chapter 3 discusses how a structure can be optimized by considering its size, shape and topology aspects.
2. The second objective was the evaluation of available meta-heuristic algorithms which can be used to solve complex single- and multi-objective optimization problems, and to use this evaluation to select an algorithm for use in the framework implementation of this study. This objective is met through the comprehensive overview of four popular optimization algorithms in chapter 4. From these, the genetic algorithm (GA) was selected for single-objective problems and its multi-objective variant, the non-dominated sorting genetic algorithm (NSGA-II), for the problems of a multi-objective nature.
3. The third objective was the development of a finite element analysis module which is required for the preceding objectives. This objective was met and is discussed in chapter 5 under section 5.1. The FEA module caters for both truss and frame elements as well as for multiple load cases. Visual and documented reporting functionality was added to these modules to simplify the validation and testing processes.
4. This objective required the development and implementation of an optimization framework which can be used for the numerical experiments in the study. This is described in chapter 5 under section 5.2. The MOEA Framework served as a starting point with much of the specialised variation operations and algorithms already available. The framework was extended to cater for structural problems as well as the addition of the GA with elitism to solve single-objective problems.
5. The fifth objective was the main objective required to achieve the first aim of this study and is discussed in chapter 6. It entailed the utilization of the software developed to achieve the previous two objectives to quantify the improvement of

the resulting truss structure by applying a more complex optimization approach. A total of four truss structures found in literature were used to gather results. The resulting weight and elapsed time for each of the seven different optimization approaches were used to determine which approach yields better results.

6. The final objective was necessary to achieve the second aim of the study. This objective, discussed in chapter 7, was to optimize the maximum displacement and weight of a frame structure as a multi-objective problem. The result from the optimization has been compared to the limit given in design codes to determine whether or not a change in this limit is justified in favour of a significant weight reduction. A total of four frame structures were used as example structures for this objective.

## 8.3 Findings

This study gathered findings regarding each of the two identified aims. The first was the quantitative comparison of optimization approaches for truss structures and the second was the multi-objective frame optimization to determine whether or not the displacement limit should be altered in favour of a more economical structure. Each of these findings are discussed in the preceding sections.

### 8.3.1 Optimization approach comparison

A total of seven different approaches towards optimizing truss structures were identified and executed on four different problems found in literature. These include the three individual approaches, size, shape and topology, along with three staged routines. The first entails topology followed by size optimization (TS), the second starts with size, followed by topology and concludes with shape optimization (STS) and the third is a topology optimization, followed by shape and concluded with size optimization (TSS). The last routine is a simultaneous (SIM) optimization routine where size, shape and topology are considered at the same time. Both the elapsed time and the resulting weight of the best structure was recorded.

From the results obtained from these experiments, it was concluded that the simultaneous approach produces, on average, a 13% lighter structure than the best staged alternative and a 22% improvement on the size-only approach. Whereas

between the individual approaches the size optimization yields a 32 % improvement compared to the shape and topology approaches.

It is worth noting that the simultaneous approach consumed more time to complete the same number of iterations which gives an indication of its increased complexity with regards to performing additional variation and encoding operations. The topology optimization was the fastest executing approach which can be attributed to its simplicity for only catering for boolean variables.

From the results obtained in this study it can be concluded that significant improvements can be made by applying a more complex optimization approach, such as considering all three aspects, namely, size shape and topology, simultaneously. It also indicates that the true optimal solution can only be found by combining the structural aspects rather than separating them as they are not independent of one-another.

### 8.3.2 Multi-objective study

Four structures were optimized in a multi-objective manner by simultaneously minimizing their maximum displacement under SLS loads and their structural weight while satisfying the design requirements of [SANS 10162-1](#). These include both two- and three-dimensional structures under different loading conditions. The optimal solution was defined as the point where the best trade-off between the two conflicting objectives is achieved.

From the obtained results it was found that up to 16 % of weight can be reduced by increasing the displacement limit, although the limit has to be increased by a large margin, rendering the structure infeasible. The defined optimum solution proved to be reasonable when its displacement is considered in relation to the limit.

When considering the position of where the limit falls on the pareto front, it excludes only a few of the solutions with a weight difference of about 20 %. This indicates that the majority of the structure's weight can be reduced before the prescribed deflection limit is reached and therefore the limit does not need to be adjusted in search of a larger weight reduction.



## 8.4 Recommendations for future research

The following recommendation can be made for future studies:

1. In the quantification study, only four truss structures were considered. This can be extended to include more test problems as well as frame structures to improve the accuracy of the obtained results.
2. The inclusion of plate or shell elements to the finite element module would enable the consideration of a more comprehensive set of structures, including floors. For this extension, the optimization of a plate element's thickness falls within the size aspect of the structure. For a concrete floor, the thickness can be considered a continuous variable with a specified range. The design module would also need to be adapted for determining the resistance of slabs.
3. In this study, a maximum of two load cases were considered where in reality a structure may be subjected to a number of loading conditions. This number may be extended to represent more realistic problems. For this extension, the optimization module needs to be altered for the maximum displacement constraints in all directions.
4. The automatic design module used in this study did not include shear capacity. The module can therefore be extended to include this check which will make the design more comprehensive.
5. This study solely focused on optimizing member sizes and shapes within a structure. The next step could be to include the optimization of connections for beam ends and supports to be either pinned or fixed, considering certain connections are more expensive than others.
6. Another extension that could be made is the optimization for fire resistance based on the protection applied. The objective could be the desired fire rating with a number of options for fireproofing different structural elements and determining the fire rating for each generated solution.
7. The optimization can also be extended to post-tensioning slabs or bridge decks. Aspects such as size, tendon profile, number of strands and cost could be optimized.

## 8.5 Concluding statement

This study compared different optimization approaches to one-another and considered whether or not the displacement limit imposed on structures should be increased in search of more economical structures. In order to answer these questions, a number of objectives were identified and met successfully with the results presented in this thesis. This study expanded the current knowledge base regarding structural optimization, however, there is still much to be learnt through future studies.

# References

- Ab Wahab, M. N., S. Nefti-Meziani, and A. Atyabi (2015). “A comprehensive review of swarm optimization algorithms”. In: *PloS one* 10.5, e0122827 (cit. on p. 45).
- Abbaspour, K., R. Schulin, and M. van Genuchten (2001). “Estimating unsaturated soil hydraulic parameters using ant colony optimization”. In: *Advances in Water Resources* 24.8, pp. 827–841. ISSN: 0309-1708. DOI: [https://doi.org/10.1016/S0309-1708\(01\)00018-5](https://doi.org/10.1016/S0309-1708(01)00018-5). URL: <http://www.sciencedirect.com/science/article/pii/S0309170801000185> (cit. on p. 51).
- Achtziger, W. (2007). “On simultaneous optimization of truss geometry and topology”. In: *Structural and Multidisciplinary Optimization* 33 (4-5). DOI: [10.1007/s00158-006-0092-0](https://doi.org/10.1007/s00158-006-0092-0) (cit. on pp. 23, 87, 88, 93).
- Ahrari, A., A. A. Atai, and K. Deb (2015). “Simultaneous topology, shape and size optimization of truss structures by fully stressed design based on evolution strategy”. In: *Engineering Optimization* 47.8, pp. 1063–1084. DOI: [10.1080/0305215x.2014.947972](https://doi.org/10.1080/0305215x.2014.947972) (cit. on pp. 23, 98).
- Ali, M. M., A. Törn, and S. Viitanen (2002). “A direct search variant of the simulated annealing algorithm for optimization involving continuous variables”. In: *Computers & Operations Research* 29.1, pp. 87–102 (cit. on p. 150).
- Andre, J., P. Siarry, and T. Dognon (2001). “An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization”. In: *Advances in Engineering Software* 32.1, pp. 49–60. ISSN: 0965-9978. DOI: [https://doi.org/10.1016/S0965-9978\(00\)00070-3](https://doi.org/10.1016/S0965-9978(00)00070-3). URL: <http://www.sciencedirect.com/science/article/pii/S0965997800000703> (cit. on p. 33).
- Ariyasingha, I. and T. Fernando (2015). “Performance analysis of the multi-objective ant colony optimization algorithms for the traveling salesman problem”. In: *Swarm and Evolutionary Computation* 23, pp. 11–26. ISSN: 2210-6502. DOI: <https://doi.org/10.1016/j.swevo.2015.02.003>. URL: <http://www.sciencedirect.com/science/article/pii/S2210650215000188> (cit. on p. 52).
- Arora, R. K. (2015). *Optimization: Algorithms and Applications*. Chapman and Hall/CRC. ISBN: 1498721125,9781498721127. URL: <http://gen.lib.rus.ec/book/index.php?md5=0AA02BE18560998A9FE2D3111BC5B8B2> (cit. on pp. 6, 9).
- Auer, B. J. (2005). “Size and shape optimization of frame and truss structures through evolutionary methods”. MA thesis. University of Idaho (cit. on pp. 23, 25).

- Bandyopadhyay, S., S. Saha, U. Maulik, and K. Deb (2008). “A simulated annealing-based multiobjective optimization algorithm: AMOSA”. In: *IEEE transactions on evolutionary computation* 12.3, pp. 269–283 (cit. on p. 40).
- Barraza, M., E. Bojórquez, E. Fernández-González, and A. Reyes-Salazar (2017). “Multi-objective optimization of structural steel buildings under earthquake loads using NSGA-II and PSO”. In: *KSCSE Journal of Civil Engineering* 21 (2). DOI: [10.1007/s12205-017-1488-7](https://doi.org/10.1007/s12205-017-1488-7) (cit. on pp. 22, 73, 114).
- Bratton, D. and J. Kennedy (2007). “Defining a Standard for Particle Swarm Optimization”. In: *Proceedings of the 2007 IEEE Swarm Intelligence Symposium*. Washington, DC, USA: IEEE Computer Society, pp. 120–127. ISBN: 1-4244-0708-7. DOI: [10.1109/SIS.2007.368035](https://doi.org/10.1109/SIS.2007.368035). URL: <http://dx.doi.org/10.1109/SIS.2007.368035> (cit. on p. 45).
- Camp, C. V. (2007). “Design of Space Trusses Using Big Bang–Big Crunch Optimization”. In: *Journal of Structural Engineering* 133 (7). DOI: [10.1061/\(asce\)0733-9445\(2007\)133:7\(999](https://doi.org/10.1061/(asce)0733-9445(2007)133:7(999) (cit. on p. 106).
- Camp, C. V. and B. J. Bichon (2004). “Design of space trusses using ant colony optimization”. In: *Journal of Structural Engineering* 130.5, pp. 741–751 (cit. on pp. 46, 87).
- Chong, E. K. and S. H. Zak (2013). *An introduction to optimization*. Vol. 76. John Wiley & Sons (cit. on pp. 1, 6, 31, 32).
- Christensen, P. W. and A. Klarbring (2008). *An introduction to structural optimization*. Vol. 153. Springer Science & Business Media (cit. on pp. 16, 18–21, 23, 24).
- Coello, C. C., M. Rudnick, and A. D. Christiansen (1994). “Using genetic algorithms for optimal design of trusses”. In: *Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on*. IEEE, pp. 88–94. DOI: [10.1109/tai.1994.346509](https://doi.org/10.1109/tai.1994.346509) (cit. on pp. 87, 98).
- Coley, D. A. (1997). *An introduction to genetic algorithms for scientists and engineers*. Har/Dis. World Scientific. ISBN: 9810236026, 9789810236021. URL: <http://gen.lib.rus.ec/book/index.php?md5=A98BA27D5B8B7AC24281043174D51033> (cit. on p. 27).
- Conn, A. R., K. Scheinberg, and L. N. Vicente (2009). *Introduction to derivative-free optimization*. Vol. 8. Siam (cit. on pp. 14–17).
- Cook, R. D., D. S. Malkus, M. E. Plesha, and R. J. Witt (2001). *Concepts and Applications of Finite Element Analysis, 4th Edition*. 4th ed. Wiley. ISBN: 0471356050 (cit. on p. 67).

- Couceiro, M. and P. Ghamisi (2015). *Fractional Order Darwinian Particle Swarm Optimization: Applications and Evaluation of an Evolutionary Algorithm*. Springer (cit. on p. 41).
- CSA S16-09: *Design of Steel Structures* (2009). Standard. CSA. ISBN: 1554912245, 9781554912247 (cit. on p. 111).
- De Jong K, A. (1975). “An Analysis of the Behavior of a Class of Genetic Adaptive Systems”. PhD thesis (cit. on pp. 27, 28).
- Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. 1st ed. Wiley. ISBN: 047187339X, 9780471873396. URL: <http://gen.lib.rus.ec/book/index.php?md5=AB91F4816A98B0377C2372BB50BCEBAF> (cit. on p. 9).
- Deb, K., S. Agrawal, A. Pratap, and T. Meyarivan (2000). “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation: NSGA-II”. In: *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*. PPSN VI. London, UK, UK: Springer-Verlag, pp. 849–858. ISBN: 3-540-41056-2. URL: <http://dl.acm.org/citation.cfm?id=645825.668937> (cit. on p. 33).
- Deb, K., M. Mohan, and S. Mishra (2003). “A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions”. In: *KanGAL report 2003002*, pp. 1–18 (cit. on p. 46).
- Degertekin S.O.; Hayalioglu, M. (2013). “Sizing truss structures using teaching-learning-based optimization”. In: *Computers & Structures* 119. DOI: [10.1016/j.compstruc.2012.12.011](https://doi.org/10.1016/j.compstruc.2012.12.011) (cit. on pp. 22, 106).
- Vo-Duy, T., D. Duong-Gia, V. Ho-Huu, H. Vu-Do, and T. Nguyen-Thoi (2017). “Multi-objective optimization of laminated composite beam structures using NSGA-II algorithm”. In: *Composite Structures* 168. Supplement C, pp. 498–509. ISSN: 0263-8223. DOI: <https://doi.org/10.1016/j.compstruct.2017.02.038>. URL: <http://www.sciencedirect.com/science/article/pii/S0263822316323728> (cit. on p. 73).
- Dyer, D. W. (2010). *The Watchmaker Framework*. URL: <http://watchmaker.uncommons.org/> (visited on 05/10/2016) (cit. on p. 74).
- Eberhart, R. C., J. Kennedy, et al. (1995). “A new optimizer using particle swarm theory”. In: *Proceedings of the sixth international symposium on micro machine and human science*. Vol. 1. New York, NY, pp. 39–43 (cit. on p. 40).
- Engel, A. (2010). *Verification, validation and testing of engineered systems*. Vol. 73. John Wiley & Sons (cit. on p. 30).

- Erbatur, O. H. F. (2002). “On efficient use of simulated annealing in complex structural optimization problems”. In: *Acta Mechanica* 157 (1-4). DOI: [10.1007/bf01182153](https://doi.org/10.1007/bf01182153) (cit. on pp. 35, 98).
- Fogel, D. B. (2005). *Defining Artificial Intelligence*. John Wiley & Sons, Inc. ISBN: 9780471749219. DOI: [10.1002/0471749214.ch1](https://doi.org/10.1002/0471749214.ch1). URL: <http://dx.doi.org/10.1002/0471749214.ch1> (cit. on p. 33).
- Gavin, H. P. (2012). *Frame Element Stiffness Matrices*. Department of Civil and Environmental Engineering Duke University. URL: <http://people.duke.edu/~hpgavin/cee421/frame-element.pdf> (cit. on p. 62).
- Gholizadeh, S. (2013). “Layout optimization of truss structures by hybridizing cellular automata and particle swarm optimization”. In: *Computers & Structures* 125. DOI: [10.1016/j.compstruc.2013.04.024](https://doi.org/10.1016/j.compstruc.2013.04.024) (cit. on p. 103).
- Goldberg, D. E. et al. (1989). *Genetic algorithms in search optimization and machine learning*. Vol. 412. Addison-wesley Reading Menlo Park (cit. on pp. 15, 26).
- Gonçalves, M. S., R. H. Lopez, and L. F. F. Miguel (2015). “Search group algorithm: A new metaheuristic method for the optimization of truss structures”. In: *Computers & Structures* 153. Supplement C, pp. 165–184. ISSN: 0045-7949. DOI: <https://doi.org/10.1016/j.compstruc.2015.03.003>. URL: <http://www.sciencedirect.com/science/article/pii/S0045794915000851> (cit. on p. 22).
- Goo, S., S. Wang, J. Hyun, and J. Jung (2016). “Topology optimization of thin plate structures with bending stress constraints”. In: *Computers & Structures* 175. Supplement C, pp. 134–143. ISSN: 0045-7949. DOI: <https://doi.org/10.1016/j.compstruc.2016.07.006>. URL: <http://www.sciencedirect.com/science/article/pii/S0045794916306101> (cit. on p. 23).
- Gordon, J. E. (1978). *Structures: or, Why things don't fall down*. Penguin Books. ISBN: 0140219617, 9780140219616. URL: <http://gen.lib.rus.ec/book/index.php?md5=4B6ACE70791BF756A14DCFBB39880504> (cit. on p. 18).
- Grefenstette, J. J. (1994). *Genetic Algorithms for Machine Learning*. 1st ed. Springer US. ISBN: 978-1-4613-6182-4, 978-1-4615-2740-4. DOI: [10.1007/978-1-4615-2740-4](https://doi.org/10.1007/978-1-4615-2740-4) (cit. on p. 33).
- Hadka, D. (2015). *MOEA Framework - A Free and Open Source Java Framework for Multiobjective Optimization. Version 2.11*. URL: <http://www.moeaframework.org> (visited on 06/10/2016) (cit. on p. 74).
- Halliday, S. (2015). *Matrix Toolkits Java*. URL: <https://github.com/fommil/matrix-toolkits-java>. (visited on 10/15/2015) (cit. on p. 68).

- Haupt, R. L. and S. E. Haupt (2004). *Practical genetic algorithms*. John Wiley & Sons (cit. on p. 35).
- He, S., Q. Wu, J. Wen, J. Saunders, and R. Paton (2004). “A particle swarm optimizer with passive congregation”. In: *Biosystems* 78.1, pp. 135–147. ISSN: 0303-2647. DOI: <http://dx.doi.org/10.1016/j.biosystems.2004.08.003> (cit. on p. 154).
- Hwaci - Applied Software Research (2017). *SQLite*. URL: [www.sqlite.org](http://www.sqlite.org) (visited on 05/11/2017) (cit. on p. 56).
- Hwang, C.-L. and A. S. M. Masud (1979). *Multiple Objective Decision Making — Methods and Applications: A State-of-the-Art Survey*. 1st ed. Lecture Notes in Economics and Mathematical Systems 164. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-09111-0, 978-3-642-45511-7 (cit. on p. 10).
- Jalili, S. and Y. Hosseinzadeh (2015). “A Cultural Algorithm for Optimal Design of Truss Structures”. In: *Latin American Journal of Solids and Structures* 12.9, pp. 1721–1747. DOI: [10.1590/1679-78251547](https://doi.org/10.1590/1679-78251547) (cit. on pp. 87, 106).
- Jamil, M. and X.-S. Yang (2013). “A literature survey of benchmark functions for global optimisation problems”. In: *International Journal of Mathematical Modelling and Numerical Optimisation* 4.2, pp. 150–194 (cit. on p. 12).
- Kaveh A.; Zolghadr, A. (2013). “Topology optimization of trusses considering static and dynamic constraints using the CSS”. In: *Applied Soft Computing* 13 (5). DOI: [10.1016/j.asoc.2012.11.014](https://doi.org/10.1016/j.asoc.2012.11.014) (cit. on pp. 88, 104).
- Kaveh, A. and S. Talatahari (2009a). “Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures”. In: *Computers & Structures* 87.5, pp. 267–283 (cit. on pp. 46, 87, 154).
- Kaveh, A. and S. Talatahari (2009b). “Size optimization of space trusses using Big Bang–Big Crunch algorithm”. In: *Computers & Structures* 87.17, pp. 1129–1140. ISSN: 0045-7949. DOI: <https://doi.org/10.1016/j.compstruc.2009.04.011>. URL: <http://www.sciencedirect.com/science/article/pii/S0045794909001394> (cit. on pp. 22, 88).
- Kaveh, A., V. R. Kalatjari, and M. H. Talebpour (2016). “Optimal Design of Steel Towers Using a Multi-Metaheuristic Based Search Method”. In: *Periodica Polytechnica Civil Engineering* 60.2, pp. 229–246. DOI: [10.3311/ppci.8222](https://doi.org/10.3311/ppci.8222) (cit. on p. 104).
- Kennedy, J. (2011). “Particle swarm optimization”. In: *Encyclopedia of machine learning*. Springer, pp. 760–766 (cit. on p. 40).
- Kiranyaz, S., T. Ince, and M. Gabbouj (2014). *Multidimensional particle swarm optimization for machine learning and pattern recognition*. Springer (cit. on p. 41).



- Kirsch, P. U. (1993). *Structural Optimization: Fundamentals and Applications*. 1st ed. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-55919-1, 978-3-642-84845-2. URL: <http://gen.lib.rus.ec/book/index.php?md5=EAD6A3F96D158F972244A2897D186AD7> (cit. on p. 18).
- Kocvara, M. and J. Zowe (1996). *How Mathematics Can Help in Design of Mechanical Structures*. Preprint 171. Institut für Angewandte Mathematik, Universität Erlangen-Nürnberg. URL: <http://www.math.fau.de/fileadmin/preprints/pr171.html> (cit. on p. 87).
- Korpus, R. (2015). *Conflicting objectives in ship design*. URL: <https://www.marinelink.com/news/conflicting-objectives402662> (visited on 04/28/2017) (cit. on p. 9).
- Koumar, A., T. Tysmans, R. Filomeno Coelho, and N. De Temmerman (2017). “An Automated Structural Optimisation Methodology for Scissor Structures Using a Genetic Algorithm”. In: *Applied Computational Intelligence and Soft Computing* 2017 (cit. on p. 73).
- Krishnakumar, K. (1990). “Micro-Genetic Algorithms For Stationary And Non-Stationary Function Optimization”. In: *1989 Symposium on Visual Communications, Image Processing, and Intelligent Robotics Systems*. Vol. 1196. International Society for Optics and Photonics, pp. 289–296. DOI: [10.1117/12.969927](https://doi.org/10.1117/12.969927). URL: <http://dx.doi.org/10.1117/12.969927> (cit. on pp. 149, 150).
- Lamberti, L. (2008). “An efficient simulated annealing algorithm for design optimization of truss structures”. In: *Computers & Structures* 86.19, pp. 1936–1953 (cit. on pp. 35, 36, 87).
- Lange, K. (2013). *Optimization*. Springer New York. URL: [http://www.ebook.de/de/product/22881142/kenneth\\_lange\\_optimization.html](http://www.ebook.de/de/product/22881142/kenneth_lange_optimization.html) (cit. on p. 13).
- Li, L., Z. Huang, F. Liu, and Q. Wu (2007). “A heuristic particle swarm optimizer for optimization of pin connected structures”. In: *Computers & Structures* 85.7, pp. 340–349 (cit. on p. 153).
- Li, M., W. Du, and F. Nian (2014). “An adaptive particle swarm optimization algorithm based on directed weighted complex network”. In: *Mathematical Problems in Engineering* 2014 (cit. on p. 45).
- Logan, D. L. (2011). *A first course in the finite element method*. SI Edition. 5th ed. Cengage Learning (cit. on pp. 58, 61, 63).
- Luh, G.-C. and C.-Y. Lin (2011). “Optimal design of truss-structures using particle swarm optimization”. In: *Computers & Structures* 89 (23-24). DOI: [10.1016/j.compstruc.2011.08.013](https://doi.org/10.1016/j.compstruc.2011.08.013) (cit. on p. 87).



- Mariaca, R. (2017). *DynamicReports*. URL: <http://www.dynamicreports.org/> (cit. on p. 71).
- McCulloch, J. (2016). *Particle Swarm Optimization*. URL: <http://mnemstudio.org/particle-swarm-introduction.htm> (visited on 07/04/2016) (cit. on p. 44).
- Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). “Equation of state calculations by fast computing machines”. In: *The journal of chemical physics* 21.6, pp. 1087–1092 (cit. on p. 37).
- Miettinen, K. (1998). *Nonlinear Multiobjective Optimization*. 1st ed. International Series in Operations Research & Management Science 12. Springer US. ISBN: 978-1-4613-7544-9, 978-1-4615-5563-6 (cit. on p. 10).
- Miguel, L. F. F., R. H. Lopez, and L. F. F. Miguel (2013). “Multimodal size, shape, and topology optimisation of truss structures using the Firefly algorithm”. In: *Advances in Engineering Software* 56, pp. 23–37. DOI: [10.1016/j.advengsoft.2012.11.006](https://doi.org/10.1016/j.advengsoft.2012.11.006) (cit. on pp. 23, 87).
- Miller, B. L. and D. E. Goldberg (1995). “Genetic algorithms, tournament selection, and the effects of noise”. In: *Complex systems* 9.3, pp. 193–212 (cit. on p. 28).
- Mitchell, M. (1999). *An introduction to genetic algorithms*. ISBN: 0-262-13316-4, 0-262-63185-7. URL: <http://gen.lib.rus.ec/book/index.php?md5=5C80C579747DFF1B6F3175615CDF2AD1> (cit. on p. 27).
- Mortazavi, A. and V. Toğan (2016). “Simultaneous size, shape, and topology optimization of truss structures using integrated particle swarm optimizer”. In: *Structural and Multidisciplinary Optimization*, pp. 1–22. DOI: [10.1007/s00158-016-1449-7](https://doi.org/10.1007/s00158-016-1449-7) (cit. on pp. 87, 88, 98, 103).
- Müller, T. E. (2015). “Development of a mathematical toolkit for finite strip models”. Unpublished final year project, Stellenbosch University (cit. on p. 68).
- Nam, D. and C. H. Park (2000). “Multiobjective simulated annealing: A comparative study to evolutionary algorithms”. In: *International Journal of Fuzzy Systems* 2.2, pp. 87–97 (cit. on p. 40).
- Nanakorn, P. and K. Meesomklin (2001). “An adaptive penalty function in genetic algorithms for structural design optimization”. In: *Computers & Structures* 79 (29–30). DOI: [10.1016/s0045-7949\(01\)00137-7](https://doi.org/10.1016/s0045-7949(01)00137-7) (cit. on p. 90).
- Nasrollahi, A. (2017). “Optimum shape of large-span trusses according to AISC-LRFD using Ranked Particles Optimization”. In: *Journal of Constructional Steel Research* 134.Supplement C, pp. 92–101. ISSN: 0143-974X. DOI: <https://doi.org/10.1016/j.jcsr.2017.03.021>. URL: <http://www.sciencedirect.com/science/article/pii/S0143974X1630712X> (cit. on p. 25).

- Patton, R. and G.-P. Liu (1994). “Robust control design via eigenstructure assignment, genetic algorithms and gradient-based optimisation”. In: *IEE Proceedings-Control Theory and Applications* 141.3, pp. 202–208 (cit. on p. 32).
- Pereda, J., M. Hoffer, and J. Pollastrini (2016). *FXyz*. <https://github.com/FXyz/FXyz> (cit. on p. 84).
- Rahami, H., A. Kaveh, and Y. Gholipour (2008). “Sizing, geometry and topology optimization of trusses via force method and genetic algorithm”. In: *Engineering Structures* 30.9, pp. 2360–2369. DOI: [10.1016/j.engstruct.2008.01.012](https://doi.org/10.1016/j.engstruct.2008.01.012) (cit. on pp. 73, 90).
- Rao, S. S. (2009). *Engineering optimization: theory and practice*. John Wiley & Sons (cit. on pp. 1, 9, 35–37, 39, 41, 42, 47, 48).
- Rios, L. M. and N. V. Sahinidis (2013). “Derivative-free optimization: a review of algorithms and comparison of software implementations”. In: *Journal of Global Optimization* 56.3, p. 1247. ISSN: 1573-2916. DOI: [10.1007/s10898-012-9951-y](https://doi.org/10.1007/s10898-012-9951-y). URL: <http://dx.doi.org/10.1007/s10898-012-9951-y> (cit. on p. 15).
- Rothlauf, F. (2011). *Design of modern heuristics: principles and application*. Springer Science & Business Media (cit. on pp. 11–14).
- Rutenbar, R. A. (1989). “Simulated annealing algorithms: An overview”. In: *IEEE Circuits and Devices Magazine* 5.1, pp. 19–26 (cit. on p. 40).
- SANS 10162-1: The structural use of steel Part 1: Limit-states design of hot-rolled steelwork* (2011). 2.01. Standard. South African Bureau of Standards. ISBN: 978-0-626-25597-8 (cit. on pp. 64, 110, 116, 118–120, 131, 135).
- Saouma, V. E. (1999). *Matrix Structural Analysis with an Introduction to Finite Elements. Lecture notes*. Dept. of Civil Environmental and Architectural Engineering University of Colorado, Boulder, CO 80309-0428. URL: [http://www4.hcmut.edu.vn/~vinhbd/Documents/Matrix%20Structural%20Ananysis%20\(with%20an%20Introduction%20to%20Finite%20Elements\).pdf](http://www4.hcmut.edu.vn/~vinhbd/Documents/Matrix%20Structural%20Ananysis%20(with%20an%20Introduction%20to%20Finite%20Elements).pdf) (cit. on p. 63).
- Schmit L.A.; Farshi, B. (1974). “Some Approximation Concepts for Structural Synthesis”. In: *AIAA Journal* 12 (5). DOI: [10.2514/3.49321](https://doi.org/10.2514/3.49321) (cit. on pp. 89, 93).
- Selvi, V. and D. R. Umarani (2010). “Comparative analysis of ant colony and particle swarm optimization techniques”. In: *International Journal of Computer Applications (0975-8887)* 5.4 (cit. on pp. 51, 52).
- Senecal, P. K. (2000). “Numerical optimization using the GEN4 micro-genetic algorithm code”. In: *University of Wisconsin-Madison* (cit. on p. 149).

- Shi, Y. and R. C. Eberhart (1998). “Parameter selection in particle swarm optimization”. In: *International Conference on Evolutionary Programming*. Springer, pp. 591–600 (cit. on p. 153).
- Sierra, M. R. and C. A. C. Coello (2005). “Improving PSO-based multi-objective optimization using crowding, mutation and e-dominance”. In: *In EMO’2005, pages 505–519. LNCS 3410*. Springer-Verlag, pp. 505–519 (cit. on p. 45).
- Simon, D. (2013). *Evolutionary Optimization Algorithms: Biologically Inspired and Population-Based Approaches to Computer Intelligence*. JOHN WILEY & SONS INC. 742 Seiten. ISBN: 0470937416. URL: [http://www.ebook.de/de/product/20253831/dan\\_simon\\_evolutionary\\_optimization\\_algorithms\\_biologically\\_inspired\\_and\\_population\\_based\\_approaches\\_to\\_computer\\_intelligence.html](http://www.ebook.de/de/product/20253831/dan_simon_evolutionary_optimization_algorithms_biologically_inspired_and_population_based_approaches_to_computer_intelligence.html) (cit. on p. 26).
- Sivakumar, P., A. Rajaraman, G. Samuel Knight, and D. Ramachandramurthy (2004). “Object-oriented optimization approach using genetic algorithms for lattice towers”. In: *Journal of computing in civil engineering* 18.2, pp. 162–171. DOI: [10.1061/\(asce\)0887-3801\(2004\)18:2\(162\)](https://doi.org/10.1061/(asce)0887-3801(2004)18:2(162)) (cit. on p. 90).
- Sonmez, F. O. (2007). “Shape optimization of 2D structures using simulated annealing”. In: *Computer methods in applied mechanics and engineering* 196.35, pp. 3279–3299 (cit. on pp. 35, 150–152).
- Southern African Institute of Steel Construction (2013). *Southern African steel construction handbook*. Southern African Institute of Steel Construction. ISBN: 9780620555111 (cit. on pp. 17, 72).
- Srinivas, N. and K. Deb (1994). “Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms”. In: *Evol. Comput.* 2.3, pp. 221–248. ISSN: 1063-6560. DOI: [10.1162/evco.1994.2.3.221](https://doi.org/10.1162/evco.1994.2.3.221). URL: <http://dx.doi.org/10.1162/evco.1994.2.3.221> (cit. on p. 33).
- Steghoefer, M. (2015). *JGAP*. URL: <https://github.com/martin-steghoefer/jgap> (visited on 05/11/2016) (cit. on p. 74).
- Tang, W., L. Tong, and Y. Gu (2005). “Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables”. In: *International Journal for Numerical Methods in Engineering* 62.13, pp. 1737–1762. DOI: [10.1002/nme.1244](https://doi.org/10.1002/nme.1244) (cit. on pp. 73, 87, 90).
- Thantulage, G. I. (2009). “Ant colony optimization based simulation of 3D automatic hose/pipe routing”. PhD thesis. Brunel University School of Engineering and Design. URL: <http://bura.brunel.ac.uk/handle/2438/4282> (cit. on p. 52).

- Toğan, V. and A. T. Daloğlu (2008). “An improved genetic algorithm with initial population strategy and self-adaptive member grouping”. In: *Computers & Structures* 86 (11-12). DOI: [10.1016/j.compstruc.2007.11.006](https://doi.org/10.1016/j.compstruc.2007.11.006) (cit. on p. 98).
- Torbaghan, M. K., S. M. Kazemi, R. Zhiani, and F. Hamed (2013). “Improved Hill Climbing and Simulated Annealing Algorithms for Size Optimization of Trusses”. In: *Proceedings of World Academy of Science, Engineering and Technology*. 74. World Academy of Science, Engineering and Technology (WASET), p. 114 (cit. on pp. 35, 36).
- Turing Finance (2016). *Using Genetic Programming to evolve Trading Strategies*. URL: <http://www.turingfinance.com/using-genetic-programming-to-evolve-security-analysis-decision-trees/> (visited on 07/05/2016) (cit. on p. 29).
- Wang, D., W. Zhang, and J. Jiang (2002). “Truss shape optimization with multiple displacement constraints”. In: *Computer Methods in Applied Mechanics and Engineering* 191.33, pp. 3597–3612. ISSN: 0045-7825. DOI: [https://doi.org/10.1016/S0045-7825\(02\)00297-9](https://doi.org/10.1016/S0045-7825(02)00297-9). URL: <http://www.sciencedirect.com/science/article/pii/S0045782502002979> (cit. on p. 25).
- Wilhelmstötter, F. (2016). *Jenetics*. URL: <http://jenetics.io/> (visited on 05/10/2016) (cit. on p. 74).
- Xia, Q., M. Y. Wang, and T. Shi (2013). “A method for shape and topology optimization of truss-like structure”. In: *Structural and Multidisciplinary Optimization* 47.5, pp. 687–697. ISSN: 1615-1488. DOI: [10.1007/s00158-012-0844-y](https://doi.org/10.1007/s00158-012-0844-y). URL: <https://doi.org/10.1007/s00158-012-0844-y> (cit. on p. 23).
- Yang, J. and Y. Zhuang (2010). “An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem”. In: *Applied Soft Computing* 10.2, pp. 653–660 (cit. on p. 154).

# Appendices

# A. Additional optimization algorithm information

Within chapter 4 various optimization algorithms which were considered for this study is discussed. This appendix chapter includes additional information regarding variations and improvements made on these algorithms.

## A.1 Micro-genetic algorithm

The simple GA (SGA) normally requires a population size of 30 to 200. A micro GA ( $\mu$ GA) requires a much smaller population of about 5 individuals, known as a  $\mu$ -population (Krishnakumar 1990). The  $\mu$ GA still follows the same approach as the normal SGA, but with a small twist.

Senecal (2000) states that mutations are not applied in the  $\mu$ GA since enough diversity is introduced after convergence of a  $\mu$ -population. Furthermore, Senecal (2000) also notes that  $\mu$ GAs reach the optimum in fewer function evaluations than a SGA, for their test functions. This provides enough motivation to consider a  $\mu$ GA for an object-oriented optimization framework.

The steps for a GA with a small population can be outlined as follows:

1. Randomly generate a small population.
2. Perform genetic operations until nominal convergence.
3. Generate a new population by transferring the best individuals of the converged population and then generate the remaining individuals randomly.
4. Go to step 2 and repeat.

Krishnakumar (1990) suggested a similar procedure for a population of 5 individuals, but where one good individual, possibly from a previous search, is intentionally inserted into the population. This procedure is the following:

1. Randomly generate a small population of 4 individuals and insert the pre-selected individual.
2. Evaluate the fitness and determine the best individual and label it. This process is termed elitism.
3. Follow a selection process to determine which individuals will be selected for crossover.
4. Apply crossover.
5. Check for nominal convergence, if converged repeat from step 1.
6. Repeat from step 2.

This start and restart procedure of the  $\mu$ GA assists in avoiding premature convergence. It is noted by Krishnakumar (1990) that a  $\mu$ GA located the optimum quickly. This may be attributed to it not having to analyse a large population.

## A.2 Direct search simulated annealing

When considering a SA and a GA, one would notice that a GA works with a population of solutions, where the SA only keeps a single solution as its best solution. Ali et al. (2002) proposed a SA approach which utilises a population of solutions which is known as the direct search simulated annealing (DSA) approach. The DSA approach was used by Sonmez (2007) for the optimization of two-dimensional trusses which yielded promising results in terms of performance.

There are mainly two differences between a normal SA and a DSA. These are listed below (Sonmez 2007):

1. The DSA keeps a set of solutions rather than just one solution.
2. The best solution is always retained by the DSA through elitism.

With this change in approach, a few adjustments need to be made to the algorithm in order to accommodate the added population of solutions. The adjustments that follow in this discussion are adapted from Sonmez (2007). Firstly, an important aspect is to

decide on a size for the population. This size, denoted by  $N$  can be determined with equation A.1. Where  $n$  denotes the dimension of the problem.

$$N = 7(n + 1) \quad (\text{A.1})$$

Secondly, it must be considered how solutions are managed within the population. Sonmez (2007) describes that, in the case a solution is accepted, the worst solution in the population is replaced.

For this procedure, methods were developed to determine the amount of iterations for the inner loop, denoted by  $L_k$ , where  $k$  refers to the  $k^{\text{th}}$  level of  $T$ . It can be determined as in equation A.2. The symbols  $f_l$  and  $f_h$  denote the lowest and highest fitness values found in the population respectively. Furthermore,  $L = 10n$ , where  $n$  is the dimension of the problem.

$$L_k = L + L(1 - e^{f_l - f_h}) \quad (\text{A.2})$$

Lastly, a different scheme for decreasing the temperature parameter is used. The factor that decreases the parameter,  $T$ , is denoted by  $\alpha$ . For the calculation for the value of  $T_{k+1}$ , equation A.3 is used. It is also important to define boundaries on this factor to ensure that it is not ridiculously large or small. The value of  $\alpha_{k+1}$  can be determined as shown in equation A.4.

$$T_{k+1} = \alpha_{k+1}T_k \quad (\text{A.3})$$

$$\alpha_{k+1} = \begin{cases} \alpha_{\max}, & \text{if } L_k > L'_k \\ \alpha_k - (\alpha_k - \alpha_{\min})(1 - L'_{k-1}/L_k), & \text{else if } L_k > L'_{k-1} \\ \alpha_{\max} - (\alpha_{\max} - \alpha_k)(L_k/L'_{k-1}), & \text{else } L_k \leq L'_{k-1} \end{cases} \quad (\text{A.4})$$

Where  $L'_k$  is the actual number of iterations executed in the  $k^{\text{th}}$  inner loop. If a better solution is not found in the inner loop,  $L'_k$  is set to  $L_k$ . If a better solution was found, the inner loop is terminated and the value of  $L'_k$  is set to the actual number of iterations.

The described procedure is graphically shown in figure A.1.



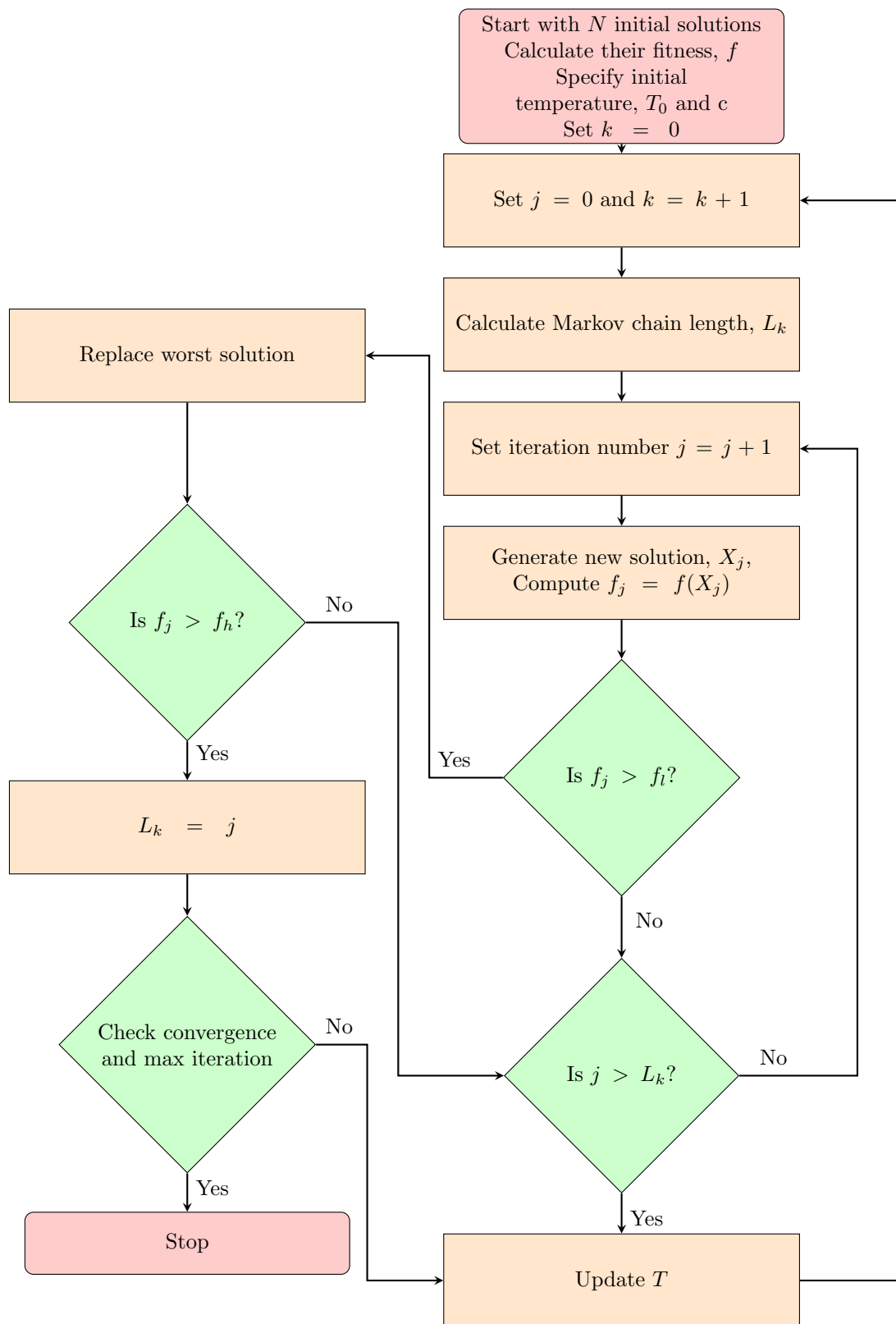


Figure A.1: The DSA process (Sonmez 2007)

### A.3 Improvements on particle swarm optimization

Since PSO was first introduced, some flaws were discovered and changes were made to the algorithm to rectify these flaws. Changes were also made to improve the overall performance of the algorithm.

The variable  $\theta$  was introduced by Shi et al. (1998) into equation 4.7 and is termed the inertia weight. This can be seen in equation A.5. The reason for this introduction is due to the fast build-up of velocities, which could lead to the best solution being skipped by the algorithm. A larger value of  $\theta$  relates to global exploration while a smaller value denotes to local search. The value of  $\theta$  also varies during the search with the assistance of equation A.6.

$$V_j(i) = \theta V_j(i-1) + c_1 r_1 [P_{best} - X_j(i-1)] + c_2 r_2 [G_{best} - X_j(i-1)] \quad (\text{A.5})$$

$$\theta(i) = \theta_{max} - \left( \frac{\theta_{max} - \theta_{min}}{i_{max}} \right) i \quad (\text{A.6})$$

Where  $\theta_{max}$  and  $\theta_{min}$  are the initial and final values of the inertia weight respectively and  $i_{max}$  is the maximum number of iteration allowed by the PSO. Commonly  $\theta_{max}$  and  $\theta_{min}$  are chosen to be 0.9 and 0.4 respectively.

Another change was made to the velocity function in order to incorporate passive congregation. Seeing as most swarms of animals congregate both actively and passively. Passive congregation can be defined as an attraction of an individual to other group members, but not a display of social behaviour (L. Li et al. 2007). Therefore this type of congregation was added to the PSO algorithm and it sometimes termed as a hybrid PSO with passive congregation (PSOPC). This change is shown in equation A.7 (L. Li et al. 2007).

$$V_j(i) = \theta V_j(i-1) + c_1 r_1 [P_{best} - X_j(i-1)] + c_2 r_2 [G_{best} - X_j(i-1)] + c_3 r_3 [R_{i-1} - X_j(i-1)] \quad (\text{A.7})$$

Where  $R_{i-1}$  is the solution of a particle that is randomly selected from the swarm.  $c_3$  and  $r_3$  is the congregation coefficient and a uniformly distributed random number

in the range of 0 to 1.

Several benchmark tests have been performed by He et al. (2004) to compare the PSO and PSOPC. The results obtained indicates that PSOPC has a better convergence rate and a higher accuracy than the normal PSO.

## **A.4 Improvements made on ant colony optimization**

Almost all the improvements made to the ACO was in the form of combining it with another optimization approach. For example, Yang et al. (2010) used a combination of a GA and an ACO to solve combinatorial optimization problems and Kaveh and Talatahari (2009a) used an ACO combined with PSO for the optimization of truss structures. Both of them obtained promising results.

## B. Truss optimization cross-section area list

### B.1 10-Bar truss

Table B.1: 10-Bar truss cross-section area list

Cross-sectional area (mm <sup>2</sup> )							
1045.16	1161.29	1283.87	1374.19	1535.48	1690.32	1696.77	1858.06
1890.32	1993.54	2019.35	2180.64	2238.71	2290.32	2341.93	2477.41
2496.77	2503.22	2696.77	2722.58	2896.77	2961.28	3096.77	3206.45
3303.22	3703.22	4658.06	5141.93	7419.34	8709.66	8967.72	9161.27
9999.98	10322.56	10903.20	12129.01	12838.68	14193.52	14774.16	17096.74
19354.80	21612.86						

### B.2 25-Bar truss

Table B.2: 25-Bar truss cross-section area list

Cross-sectional area (mm <sup>2</sup> )							
64.52	129.03	193.55	258.06	322.58	387.10	451.61	516.13
580.64	645.16	709.68	774.19	838.71	903.22	967.74	1032.26
1096.77	1161.29	1225.80	1290.32	1354.84	1419.35	1483.87	1548.38
1612.90	1677.42	1806.45	1935.48	2064.51	2193.54		

## B.3 47-Bar truss

Table B.3: 47-Bar truss cross-section area list

Cross-sectional area (mm <sup>2</sup> )							
64.52	129.03	193.55	258.06	322.58	387.10	451.61	516.13
580.64	645.16	709.68	774.19	838.71	903.22	967.74	1032.26
1096.77	1161.29	1225.80	1290.32	1354.84	1419.35	1483.87	1548.38
1612.90	1677.42	1741.93	1806.45	1870.96	1935.48	2000.00	2064.51
2129.03	2193.54	2258.06	2322.58	2387.09	2451.61	2516.12	2580.64
2645.16	2709.67	2774.19	2838.70	2903.22	2967.74	3032.25	3096.77
3161.28	3225.80						

## B.4 72-Bar truss

Table B.4: 72-Bar truss cross-section area list

Cross-sectional area (mm <sup>2</sup> )						
71.61	90.97	126.45	161.29	198.06	252.26	285.16
363.23	388.39	494.19	506.45	641.29	645.16	792.26
816.77	940.00	1008.39	1045.16	1161.29	1283.87	1374.19
1535.48	1690.32	1696.77	1858.06	1890.32	1993.54	729.03
2180.64	2238.70	2290.32	2341.93	2477.41	2496.77	2503.22
2696.77	2722.58	2896.77	2961.28	3096.77	3206.45	3303.22
3703.22	4658.06	5141.93	5503.21	5999.99	6999.99	7419.43
8709.66	8967.72	9161.27	9999.98	10322.56	10903.20	12129.01
12838.68	14193.52	14774.16	15806.42	17096.74	18064.48	19354.80
21612.86						

## C. Analysis report example



## Input Data :

### Nodes :

Name	X	Y	Z
n1	0.0	0.0	0.0
sn1_e1	0.0	500.0E-3	0.0
sn2_e1	0.0	1.0	0.0
sn3_e1	0.0	1.5	0.0
n2	0.0	2.0	0.0
sn1_e2	750.0E-3	2.3	0.0
sn2_e2	1.5	2.5	0.0
sn3_e2	2.3	2.8	0.0
n3	3.0	3.0	0.0

### Elements :

Name	From	To	Section	Local rotation	End fixity
se1_e1	n1	sn1_e1	254x254x89	0.0	Fully fixed
se2_e1	sn1_e1	sn2_e1	254x254x89	0.0	Fully fixed
se3_e1	sn2_e1	sn3_e1	254x254x89	0.0	Fully fixed
se4_e1	sn3_e1	n2	254x254x89	0.0	Fully fixed
se1_e2	n2	sn1_e2	203x133x30	0.0	Fully fixed
se2_e2	sn1_e2	sn2_e2	203x133x30	0.0	Fully fixed
se3_e2	sn2_e2	sn3_e2	203x133x30	0.0	Fully fixed
se4_e2	sn3_e2	n3	203x133x30	0.0	Fully fixed

### Element loads :

LoadCase	Element	Local direction	Magnitude
LIVE	se4_e2	2	3.0E3
LIVE	se2_e2	2	3.0E3
LIVE	se1_e2	2	3.0E3
LIVE	se3_e2	2	3.0E3
DEAD	se3_e2	2	1.0E3
DEAD	se1_e2	2	1.0E3
DEAD	se2_e2	2	1.0E3
DEAD	se4_e2	2	1.0E3

### Supports :

Node	Restraint
n1	Pinned

Node	Restraint
n3	Fixed

**Load Combinations :**

Name	LocaCase	Factor
ULS	LIVE	1.60
ULS	DEAD	1.20

**Results :****Reactions :**

	Node	Rx	Ry	Rz	Mx	My	Mz
<b>ULS</b>							
	n1	2.07E3	10.25E3	0.00	0.00	0.00	0.00
	n3	-8.07E3	7.75E3	-5.46E3	0.00	0.00	0.00

**Nodal displacements :**

	Node	dx	dy	dz	phix	phiy	phiz
<b>ULS</b>							
	n1	0.00	0.00	0.00	0.00	0.00	35.42E-6
	sn1_e1	-16.20E-6	-2.25E-6	0.00	0.00	0.00	26.37E-6
	sn2_e1	-23.35E-6	-4.50E-6	0.00	0.00	0.00	-787.86E-9
	sn3_e1	-12.40E-6	-6.74E-6	0.00	0.00	0.00	-46.05E-6
	n2	25.71E-6	-8.99E-6	0.00	0.00	0.00	-109.41E-6
	sn1_e2	83.25E-6	-198.64E-6	0.00	0.00	0.00	-271.04E-6
	sn2_e2	112.02E-6	-301.98E-6	0.00	0.00	0.00	35.26E-6
	sn3_e2	59.15E-6	-160.41E-6	0.00	0.00	0.00	296.58E-6
	n3	0.00	0.00	0.00	0.00	0.00	0.00
	Max	112.02E-6	301.98E-6	0.00	0.00	0.00	296.58E-6

**Element Forces :**

	Element						
	Node 1	Fx	Fy	Fz	Mx	My	Mz
	Node 2	Fx	Fy	Fz	Mx	My	Mz
<b>ULS</b>							
	se1_e1						
	n1	10.25E3	-2.07E3	0.00	0.00	0.00	0.00
	sn1_e1	-10.25E3	2.07E3	0.00	0.00	0.00	-1.04E3
	se2_e1						
	sn1_e1	10.25E3	-2.07E3	0.00	0.00	0.00	1.04E3
	sn2_e1	-10.25E3	2.07E3	0.00	0.00	0.00	-2.07E3
	se3_e1						
	sn2_e1	10.25E3	-2.07E3	0.00	0.00	0.00	2.07E3
	sn3_e1	-10.25E3	2.07E3	0.00	0.00	0.00	-3.11E3



	Element						
	Node 1	Fx	Fy	Fz	Mx	My	Mz
	Node 2	Fx	Fy	Fz	Mx	My	Mz
se4_e1							
sn3_e1		10.25E3	-2.07E3	0.00	0.00	0.00	3.11E3
n2		-10.25E3	2.07E3	0.00	0.00	0.00	-4.14E3
se1_e2							
n2		5.21E3	9.07E3	0.00	0.00	0.00	4.14E3
sn1_e2		-5.21E3	-4.33E3	0.00	0.00	0.00	1.15E3
se2_e2							
sn1_e2		5.21E3	4.33E3	0.00	0.00	0.00	-1.15E3
sn2_e2		-5.21E3	416.00	0.00	0.00	0.00	2.70E3
se3_e2							
sn2_e2		5.21E3	-416.00	0.00	0.00	0.00	-2.70E3
sn3_e2		-5.21E3	5.16E3	0.00	0.00	0.00	496.11
se4_e2							
sn3_e2		5.21E3	-5.16E3	0.00	0.00	0.00	-496.11
n3		-5.21E3	9.90E3	0.00	0.00	0.00	-5.46E3

### Analysis statistics :

Elapsed time : 6.43 [ms]

Number of DOFs : 27

Number of Nodes : 9

Number of Elements : 8

Number of Load Cases : 2

Number of Load Combinations : 1

## D. Optimization report example



Designer	Optimizer
Date	10/07/2017
Project	Masters

Page	1
Total	2

## Optimization report:

Population size: 80

Function evaluation: 100000

## Size Settings

### Variable elements

#### e1

356x171x45	356x171x67	457x191x67	457x191x89	254x146x31	406x178x67	254x146x37	457x191x82
IPE-AA180	203x133x25	IPE-AA140	IPE-AA160	305x165x54	406x140x46	533x210x92	305x102x33
356x171x57	IPE120	IPE-AA200	IPE-AA100	406x178x54	IPE100	533x210x109	IPE-AA120
254x146x43	406x178x74	457x191x98	IPE200	305x102x28	IPE180	305x102x25	533x210x101
406x140x39	IPE160	305x165x40	406x178x60	457x191x74	IPE140	356x171x51	305x165x46
533x210x122	203x133x30	533x210x82	203x203x86	305x305x137	305x305x158	203x203x60	203x203x71
254x254x107	305x305x198	203x203x52	254x254x73	254x254x167	203x203x46	254x254x89	254x254x132
305x305x118	152x152x30	152x152x23	152x152x37	305x305x97	305x305x240		

#### e5

356x171x45	356x171x67	457x191x67	457x191x89	254x146x31	406x178x67	254x146x37	457x191x82
IPE-AA180	203x133x25	IPE-AA140	IPE-AA160	305x165x54	406x140x46	533x210x92	305x102x33
356x171x57	IPE120	IPE-AA200	IPE-AA100	406x178x54	IPE100	533x210x109	IPE-AA120
254x146x43	406x178x74	457x191x98	IPE200	305x102x28	IPE180	305x102x25	533x210x101
406x140x39	IPE160	305x165x40	406x178x60	457x191x74	IPE140	356x171x51	305x165x46
533x210x122	203x133x30	533x210x82	203x203x86	305x305x137	305x305x158	203x203x60	203x203x71
254x254x107	305x305x198	203x203x52	254x254x73	254x254x167	203x203x46	254x254x89	254x254x132
305x305x118	152x152x30	152x152x23	152x152x37	305x305x97	305x305x240		

#### e13

356x171x45	356x171x67	457x191x67	457x191x89	254x146x31	406x178x67	254x146x37	457x191x82
IPE-AA180	203x133x25	IPE-AA140	IPE-AA160	305x165x54	406x140x46	533x210x92	305x102x33
356x171x57	IPE120	IPE-AA200	IPE-AA100	406x178x54	IPE100	533x210x109	IPE-AA120
254x146x43	406x178x74	457x191x98	IPE200	305x102x28	IPE180	305x102x25	533x210x101
406x140x39	IPE160	305x165x40	406x178x60	457x191x74	IPE140	356x171x51	305x165x46
533x210x122	203x133x30	533x210x82	203x203x86	305x305x137	305x305x158	203x203x60	203x203x71
254x254x107	305x305x198	203x203x52	254x254x73	254x254x167	203x203x46	254x254x89	254x254x132
305x305x118	152x152x30	152x152x23	152x152x37	305x305x97	305x305x240		

#### e17

356x171x45	356x171x67	457x191x67	457x191x89	254x146x31	406x178x67	254x146x37	457x191x82
IPE-AA180	203x133x25	IPE-AA140	IPE-AA160	305x165x54	406x140x46	533x210x92	305x102x33
356x171x57	IPE120	IPE-AA200	IPE-AA100	406x178x54	IPE100	533x210x109	IPE-AA120
254x146x43	406x178x74	457x191x98	IPE200	305x102x28	IPE180	305x102x25	533x210x101
406x140x39	IPE160	305x165x40	406x178x60	457x191x74	IPE140	356x171x51	305x165x46



Designer	Optimizer
Date	10/07/2017
Project	Masters

Page	2
Total	2

533x210x122 203x133x30 533x210x82 203x203x86 305x305x137 305x305x158 203x203x60 203x203x71  
 254x254x107 305x305x198 203x203x52 254x254x73 254x254x167 203x203x46 254x254x89 254x254x132  
 305x305x118 152x152x30 152x152x23 152x152x37 305x305x97 305x305x240

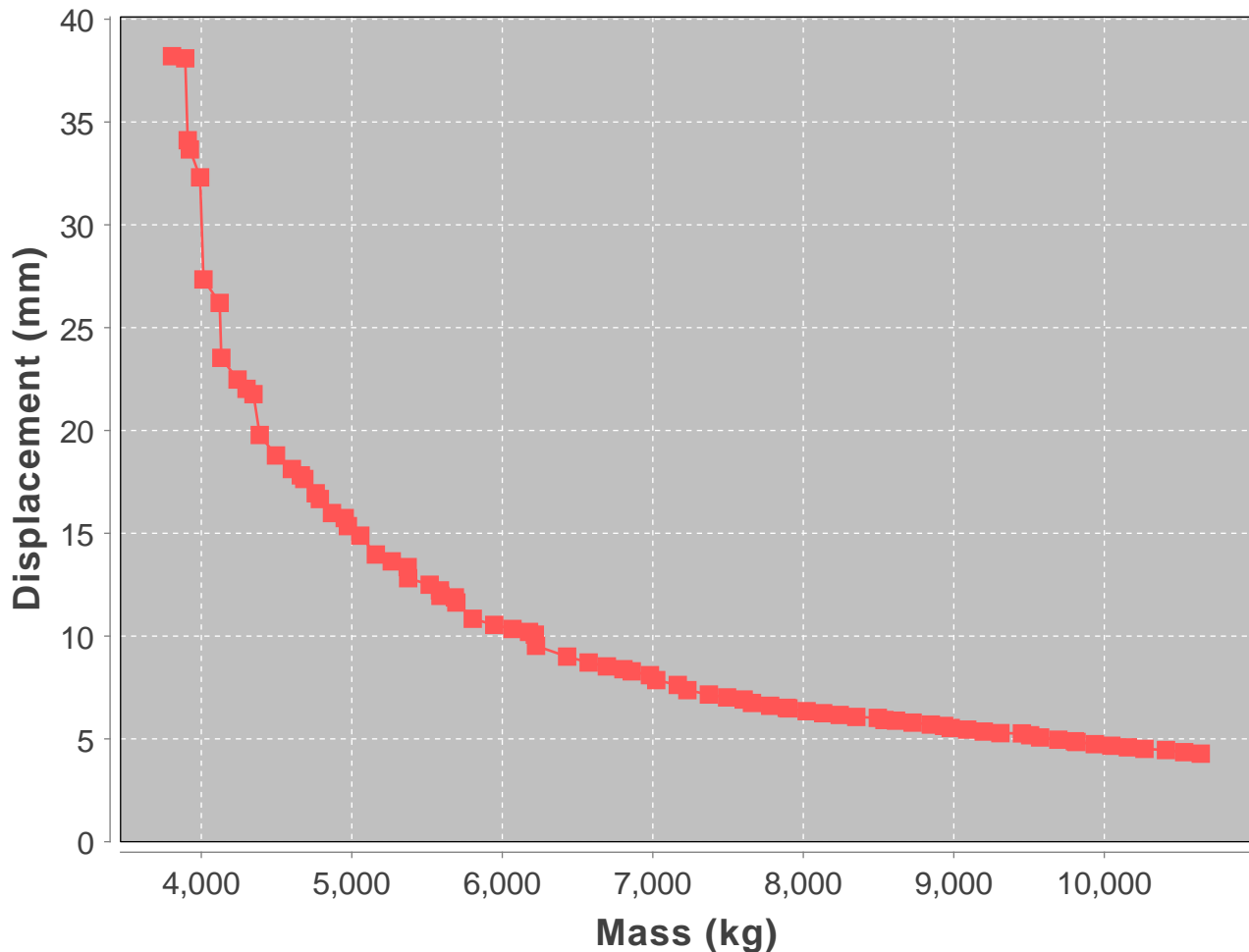
### Grouping elements

e6 e7 e8 e5  
 e15 e14 e16 e13  
 e19 e18 e20 e17  
 e2 e3 e4 e9 e10 e11 e12 e1

### Post optimization:

Total time = 64.76E3 ms

### Result objectives



## E. Design report example



Designer	Auto
Date	10/07/2017
Project	Masters

Page	1
Total	11

\* Disclaimer: Bending is ignored for members with  $M_u < 3.50 \text{ kNm}$

## Designing for LoadCase ULS

### Designing member e1 as a Beam-Column

Section : 254x254x89

Forces: Axial = 10.3 kN  $M_{ux} = 4.14 \text{ kNm}$   $M_{uy} = 0.00 \text{ kNm}$

### Overall member Strength

### Classify Bending section:

#### Flange:

$$\frac{b}{2 \cdot t_f} \leq \frac{145}{\sqrt{f_y}}$$

$$\frac{0.26}{2 \cdot 17.30 \cdot 10^{-3}} \leq \frac{145}{\sqrt{355}}$$

$$7.40 \leq 7.70 \text{ Class 1}$$

SANS 10162-1 13.8.2  
b)

SANS 10162-1 Table 4

#### Web:

$$\frac{h - 2 \cdot t_f}{t_w} \leq \frac{1100}{\sqrt{f_y}} \cdot \left( 1 - 0.39 \cdot \frac{C_u}{\phi \cdot C_y} \right)$$

$$\frac{0.26 - 2 \cdot 17.30 \cdot 10^{-3}}{10.50 \cdot 10^{-3}} \leq \frac{1100}{\sqrt{355}} \cdot \left( 1 - 0.39 \cdot \frac{10.25 \cdot 10^3}{0.9 \cdot 4.05 \cdot 10^6} \right)$$

$$21.5 \leq 58.3 \text{ Class 1}$$

SANS 10162-1 Table 4

### Class 1 in bending

$$f_{ex} = \frac{\pi^2 \cdot E}{\left( \frac{K_x \cdot L_x}{r_x} \right)^2}$$

$$f_{ex} = \frac{\pi^2 \cdot 200 \cdot 10^9}{\left( \frac{1.0 \cdot 2.0}{0.11} \right)^2}$$

$$f_{ex} = 6190 \text{ MPa}$$

SANS 10162-1 13.3.2

### Check Slenderness

$$\frac{K_x \cdot L_x}{r_x} \leq 200$$

$$\frac{1.0 \cdot 2.0}{0.11} \leq 200$$

$$17.9 \leq 200 \text{ OK}$$

SANS 10162-1 10.4.2

$$\frac{K_y \cdot L_y}{r_y} \leq 200$$

$$\frac{1.0 \cdot 2.0}{65.20 \cdot 10^{-3}} \leq 200$$

$$30.7 \leq 200 \text{ OK}$$

### Classify I or H for compression

#### Flange:



Designer	Auto
Date	10/07/2017
Project	Masters

Page	2
Total	11

$$\frac{b}{2 \cdot t_f} \leq \frac{200}{\sqrt{f_y}}$$

$$\frac{0.26}{2 \cdot 17.30 \cdot 10^{-3}} \leq \frac{200}{\sqrt{355}}$$

$$7.40 \leq 10.6 \text{ Class 3}$$

SANS 10162-1 Table 3

**Web:**

$$\frac{h - 2 \cdot t_f}{t_w} \leq \frac{670}{\sqrt{f_y}}$$

$$\frac{h - 2 \cdot 17.30 \cdot 10^{-3}}{10.50 \cdot 10^{-3}} \leq \frac{670}{\sqrt{355}}$$

$$21.5 \leq 35.6 \text{ Class 3}$$

SANS 10162-1 Table 3

**Class 3 in Compression**

$$\lambda_x = \sqrt{\frac{f_y}{f_{ex}}}$$

$$\lambda_x = \sqrt{\frac{355 \cdot 10^6}{6.19 \cdot 10^9}}$$

$$\lambda_x = 0.24$$

SANS 10162-1 13.3.1

$$C_{rx} = \phi \cdot A \cdot f_y \cdot (1 + \lambda_x^{2.0})^{-1/2.0}$$

$$C_{rx} = 0.9 \cdot 11.40 \cdot 10^{-3} \cdot 355 \cdot 10^6 \cdot (1 + 0.24^{2.134})^{-1/2.134}$$

$$C_{rx} = 3584 \text{ kN}$$

SANS 10162-1 13.3.1

$$\omega_2 = \frac{4 \cdot M_{max}}{\sqrt{M_{max}^2 + 4 \cdot (M_a)^2 + 7 \cdot (M_b)^2 + 4 \cdot (M_c)^2}}$$

$$4 \cdot 4.14 \cdot 10^3$$

$$\omega_2 = \frac{4 \cdot 4.14 \cdot 10^3}{\sqrt{4.14 \cdot 10^3^2 + 4 \cdot (1.04 \cdot 10^3)^2 + 7 \cdot (2.07 \cdot 10^3)^2 + 4 \cdot (3.11 \cdot 10^3)^2}}$$

$$\omega_2 = 1.75$$

$$\omega_2 \leq 2.5 \text{ OK}$$

CSA S16-09 13.6

$$M_{rx} = \phi \cdot Z_{plx} \cdot f_y$$

$$M_{rx} = 0.9 \cdot 1.22 \cdot 10^{-3} \cdot 355 \cdot 10^6$$

$$M_{rx} = 391 \text{ kNm}$$

SANS 10162-1 13.5

$$M_{ry} = \phi \cdot Z_{ply} \cdot f_y$$

$$M_{ry} = 0.9 \cdot 574 \cdot 10^{-6} \cdot 355 \cdot 10^6$$

$$M_{ry} = 183 \text{ kNm}$$

SANS 10162-1 13.5

$$\frac{C_u}{C_r} + \frac{\alpha \cdot U_{1x} \cdot M_{ux}}{M_{rx}} + \frac{\beta \cdot U_{1y} \cdot M_{uy}}{M_{ry}} \leq 1.0$$

$$\frac{10.25 \cdot 10^3}{3.58 \cdot 10^6} + \frac{1.0 \cdot 1.0 \cdot 4.14 \cdot 10^3}{391.39 \cdot 10^3} + \frac{1.0 \cdot 1.0 \cdot 0}{183.39 \cdot 10^3} \leq 1.0$$

$$0.01 \leq 1.0 \text{ OK}$$

SANS 10162-1 13.8

**Lateral torsional buckling Strength**SANS 10162-1 13.8.2  
c)



Designer	Auto
Date	10/07/2017
Project	Masters

Page	3
Total	11

$$f_{ey} = \frac{\pi^2 \cdot E}{\left(\frac{K_y \cdot L_y}{r_y}\right)^2}$$

$$f_{ey} = \frac{\pi^2 \cdot 200 \cdot 10^9}{\left(\frac{1.0 \cdot 2.0}{65.20 \cdot 10^{-3}}\right)^2}$$

$$f_{ey} = 2098 \text{ MPa}$$

$$\lambda_y = \sqrt{\frac{f_y}{f_{ey}}}$$

$$\lambda_y = \sqrt{\frac{355 \cdot 10^6}{2.10 \cdot 10^9}}$$

$$\lambda_y = 0.41$$

$$C_{ry} = \phi \cdot A \cdot f_y \cdot (1 + \lambda_y^{2 \cdot n})^{-1/n}$$

$$C_{ry} = 0.9 \cdot 11.40 \cdot 10^{-3} \cdot 355 \cdot 10^6 \cdot (1 + 0.41^{2 \cdot 1.34})^{-1/1.34}$$

$$C_{ry} = 3410 \text{ kN}$$

$$\bar{r}_o^2 = (x_o)^2 + (y_o)^2 + (r_x)^2 + (r_y)^2$$

$$\bar{r}_o^2 = (0)^2 + (0)^2 + (0.11)^2 + (65.20 \cdot 10^{-3})^2$$

$$\bar{r}_o^2 = 16795 \text{ mm}^2$$

$$f_{ez} = \left( \frac{\pi^2 \cdot E \cdot C_w}{K_z^2 \cdot L_z^2} + G \cdot J \right) \cdot \frac{1}{A \cdot \bar{r}_o^2}$$

$$f_{ez} = \left( \frac{\pi^2 \cdot 200 \cdot 10^9 \cdot 714 \cdot 10^{-9}}{1.02 \cdot 2.0^2} + 77 \cdot 10^9 \cdot 1.04 \cdot 10^{-6} \right) \cdot \frac{1}{11.40 \cdot 10^{-3} \cdot 16.80 \cdot 10^{-3}}$$

$$f_{ez} = 2257 \text{ MPa}$$

$$\lambda_{ez} = \sqrt{\frac{f_y}{f_{ez}}}$$

$$\lambda_{ez} = \sqrt{\frac{355 \cdot 10^6}{2.26 \cdot 10^9}}$$

$$\lambda_{ez} = 0.40$$

$$C_{rez} = \phi \cdot A \cdot f_y \cdot (1 + \lambda_{ez}^{2 \cdot n})^{-1/n}$$

$$C_{rez} = 0.9 \cdot 11.40 \cdot 10^{-3} \cdot 355 \cdot 10^6 \cdot (1 + 0.40^{2 \cdot 1.34})^{-1/1.34}$$

$$C_{rez} = 3430 \text{ kN}$$

SANS 10162-1 13.3.2

SANS 10162-1 13.3.1

SANS 10162-1 13.3.1

SANS 10162-1 13.3.2

SANS 10162-1 13.3.2

SANS 10162-1 13.3.1

SANS 10162-1 13.3.1





Designer	Auto
Date	10/07/2017
Project	Masters

Page	4
Total	11

$$M'_{cr} = \frac{\pi}{K \cdot L} \cdot \sqrt{E \cdot I_y \cdot G \cdot J + \left( \frac{\pi \cdot E}{K \cdot L} \right)^2 \cdot I_y \cdot C_w}$$

$$M'_{cr} = \frac{\pi}{1.0 \cdot 2.0} \cdot$$

$$\sqrt{200 \cdot 10^9 \cdot 48.30 \cdot 10^{-6} \cdot 77 \cdot 10^9 \cdot 1.04 \cdot 10^{-6} + \left( \frac{\pi \cdot 200 \cdot 10^9}{1.0 \cdot 2.0} \right)^2 \cdot 48.30 \cdot 10^{-6} \cdot 714 \cdot 10^{-9}}$$

$$M'_{cr} = 3209 \text{ kNm}$$

$$M_{cr} = \omega_2 \cdot M'_{cr}$$

$$M_{cr} = 1.75 \cdot 3.21 \cdot 10^6$$

$$M_{cr} = 5603 \text{ kNm}$$

$$M_{cr} > 0.67 \cdot M_p$$

$$M_{cr} > 0.67 \cdot 391.39 \cdot 10^3$$

$$M_{cr} > 262 \text{ kNm}$$

$$M_r = 1.15 \cdot \phi \cdot M_p \cdot \left( 1 - \frac{0.28 \cdot M_p}{M_{cr}} \right)$$

$$M_r = 1.15 \cdot 0.9 \cdot 391.39 \cdot 10^3 \cdot \left( 1 - \frac{0.28 \cdot 391.39 \cdot 10^3}{5.60 \cdot 10^6} \right)$$

$$M_r = 397 \text{ kNm}$$

$$M_r \leq \phi \cdot M_p$$

$$M_r \leq 0.9 \cdot 391.39 \cdot 10^3$$

$$M_r \leq 352.25 \cdot 10^3$$

NOT OK

$$M_r = \phi \cdot M_p$$

$$M_r = 352 \text{ kNm}$$

$$M_{ry} = \phi \cdot Z_{ply} \cdot f_y$$

$$M_{ry} = 0.9 \cdot 574 \cdot 10^{-6} \cdot 355 \cdot 10^6$$

$$M_{ry} = 183 \text{ kNm}$$

$$\frac{C_u}{C_r} + \frac{\alpha \cdot U_{1x} \cdot M_{ux}}{M_{rx}} + \frac{\beta \cdot U_{1y} \cdot M_{uy}}{M_{ry}} \leq 1.0$$

$$\frac{10.25 \cdot 10^3}{3.41 \cdot 10^6} + \frac{1.0 \cdot 1.0 \cdot 4.14 \cdot 10^3}{352.25 \cdot 10^3} + \frac{1.0 \cdot 1.0 \cdot 0}{183.39 \cdot 10^3} \leq 1.0$$

$$0.01 \leq 1.0 \text{ OK}$$

$$M'_{cr} = \frac{\pi}{K \cdot L} \cdot \sqrt{E \cdot I_y \cdot G \cdot J + \left( \frac{\pi \cdot E}{K \cdot L} \right)^2 \cdot I_y \cdot C_w}$$

$$M'_{cr} = \frac{\pi}{1.0 \cdot 2.0} \cdot$$

$$\sqrt{200 \cdot 10^9 \cdot 48.30 \cdot 10^{-6} \cdot 77 \cdot 10^9 \cdot 1.04 \cdot 10^{-6} + \left( \frac{\pi \cdot 200 \cdot 10^9}{1.0 \cdot 2.0} \right)^2 \cdot 48.30 \cdot 10^{-6} \cdot 714 \cdot 10^{-9}}$$

$$M'_{cr} = 3209 \text{ kNm}$$

$$M_{cr} = \omega_2 \cdot M'_{cr}$$

$$M_{cr} = 1.75 \cdot 3.21 \cdot 10^6$$

$$M_{cr} = 5603 \text{ kNm}$$

SANS 10162-1 13.6

SANS 10162-1 13.6

SANS 10162-1 13.5

SANS 10162-1 13.8

SANS 10162-1 13.6



Designer	Auto
Date	10/07/2017
Project	Masters

Page	5
Total	11

$$M_{cr} > 0.67 \cdot M_p$$

$$M_{cr} > 0.67 \cdot 391.39 \cdot 10^3$$

$$M_{cr} > 262 \text{ kNm}$$

$$M_r = 1.15 \cdot \phi \cdot M_p \cdot \left(1 - \frac{0.28 \cdot M_p}{M_{cr}}\right)$$

$$M_r = 1.15 \cdot 0.9 \cdot 391.39 \cdot 10^3 \cdot \left(1 - \frac{0.28 \cdot 391.39 \cdot 10^3}{5.60 \cdot 10^6}\right)$$

$$M_r = 397 \text{ kNm}$$

$$M_r \leq \phi \cdot M_p$$

$$M_r \leq 0.9 \cdot 391.39 \cdot 10^3$$

$$M_r \leq 352.25 \cdot 10^3$$

NOT OK

$$M_r = \phi \cdot M_p$$

$$M_r = 352 \text{ kNm}$$

$$M_{ry} = \phi \cdot Z_{ply} \cdot f_y$$

$$M_{ry} = 0.9 \cdot 574 \cdot 10^{-6} \cdot 355 \cdot 10^6$$

$$M_{ry} = 183 \text{ kNm}$$

$$\frac{M_{ux}}{M_{rx}} + \frac{M_{uy}}{M_{ry}} \leq 1.0$$

$$\frac{4.14 \cdot 10^3}{352.25 \cdot 10^3} + \frac{0}{183.39 \cdot 10^3} \leq 1.0$$

$$0.01 \leq 1.0 \text{ OK}$$

SANS 10162-1 13.6

SANS 10162-1 13.5

SANS 10162-1 13.8

**Member e1 has sufficient capacity**



Designer	Auto
Date	10/07/2017
Project	Masters

Page	6
Total	11

## Designing member e2 as a Beam-Column

Section : 203x133x30

Forces: Axial = 5.21 kN Mux = 5.46 kNm Muy = 0.00 kNm

### Overall member Strength

### Classify Bending section:

#### Flange:

$$\frac{b}{2 \cdot t_f} \leq \frac{145}{\sqrt{f_y}}$$

$$\frac{0.13}{2 \cdot 9.60 \cdot 10^{-3}} \leq \frac{145}{\sqrt{355}}$$

$$6.97 \leq 7.70 \text{ Class 1}$$

#### Web:

$$\frac{h - 2 \cdot t_f}{t_w} \leq \frac{1100}{\sqrt{f_y}} \cdot \left( 1 - 0.39 \cdot \frac{C_u}{\phi \cdot C_y} \right)$$

$$\frac{0.21 - 2 \cdot 9.60 \cdot 10^{-3}}{6.40 \cdot 10^{-3}} \leq \frac{1100}{\sqrt{355}} \cdot \left( 1 - 0.39 \cdot \frac{5.21 \cdot 10^3}{0.9 \cdot 1.36 \cdot 10^6} \right)$$

$$29.3 \leq 58.3 \text{ Class 1}$$

#### Class 1 in bending

$$f_{ex} = \frac{\pi^2 \cdot E}{\left( \frac{K_x \cdot L_x}{r_x} \right)^2}$$

$$f_{ex} = \frac{\pi^2 \cdot 200 \cdot 10^9}{\left( \frac{1.0 \cdot 3.16}{87 \cdot 10^{-3}} \right)^2}$$

$$f_{ex} = 1494 \text{ MPa}$$

#### Check Slenderness

$$\frac{K_x \cdot L_x}{r_x} \leq 200$$

$$\frac{1.0 \cdot 3.16}{87 \cdot 10^{-3}} \leq 200$$

$$36.3 \leq 200 \text{ OK}$$

$$\frac{K_y \cdot L_y}{r_y} \leq 200$$

$$\frac{1.0 \cdot 3.16}{31.70 \cdot 10^{-3}} \leq 200$$

$$99.8 \leq 200 \text{ OK}$$

### Classify I or H for compression

#### Flange:

SANS 10162-1 13.8.2  
b)

SANS 10162-1 Table 4

SANS 10162-1 Table 4

SANS 10162-1 13.3.2

SANS 10162-1 10.4.2



Designer	Auto
Date	10/07/2017
Project	Masters

Page	7
Total	11

$$\frac{b}{2 \cdot t_f} \leq \frac{200}{\sqrt{f_y}}$$

$$\frac{0.13}{2 \cdot 9.60 \cdot 10^{-3}} \leq \frac{200}{\sqrt{355}}$$

$$6.97 \leq 10.6 \text{ Class 3}$$

SANS 10162-1 Table 3

**Web:**

$$\frac{h - 2 \cdot t_f}{t_w} \leq \frac{670}{\sqrt{f_y}}$$

$$\frac{h - 2 \cdot 9.60 \cdot 10^{-3}}{6.40 \cdot 10^{-3}} \leq \frac{670}{\sqrt{355}}$$

$$29.3 \leq 35.6 \text{ Class 3}$$

SANS 10162-1 Table 3

**Class 3 in Compression**

$$\lambda_x = \sqrt{\frac{f_y}{f_{ex}}}$$

$$\lambda_x = \sqrt{\frac{355 \cdot 10^6}{1.49 \cdot 10^9}}$$

$$\lambda_x = 0.49$$

SANS 10162-1 13.3.1

$$C_{rx} = \phi \cdot A \cdot f_y \cdot (1 + \lambda_x^{2n})^{-1/n}$$

$$C_{rx} = 0.9 \cdot 3.82 \cdot 10^{-3} \cdot 355 \cdot 10^6 \cdot (1 + 0.49^{2 \cdot 1.34})^{-1/1.34}$$

$$C_{rx} = 1103 \text{ kN}$$

SANS 10162-1 13.3.1

$$\omega_2 = \frac{4 \cdot M_{max}}{\sqrt{M_{max}^2 + 4 \cdot (M_a)^2 + 7 \cdot (M_b)^2 + 4 \cdot (M_c)^2}}$$

$$4 \cdot 5.46 \cdot 10^3$$

$$\omega_2 = \frac{4 \cdot 5.46 \cdot 10^3}{\sqrt{5.46 \cdot 10^3 + 4 \cdot (1.15 \cdot 10^3)^2 + 7 \cdot (2.70 \cdot 10^3)^2 + 4 \cdot (496.11)^2}}$$

$$\omega_2 = 2.34$$

$$\omega_2 \leq 2.5 \text{ OK}$$

CSA S16-09 13.6

$$M_{rx} = \phi \cdot Z_{plx} \cdot f_y$$

$$M_{rx} = 0.9 \cdot 314 \cdot 10^{-6} \cdot 355 \cdot 10^6$$

$$M_{rx} = 100 \text{ kNm}$$

SANS 10162-1 13.5

$$M_{ry} = \phi \cdot Z_{ply} \cdot f_y$$

$$M_{ry} = 0.9 \cdot 88.10 \cdot 10^{-6} \cdot 355 \cdot 10^6$$

$$M_{ry} = 28.1 \text{ kNm}$$

SANS 10162-1 13.5

$$f_{ey} = \frac{\pi^2 \cdot E}{\left(\frac{K_y L_y}{r_y}\right)^2}$$

$$f_{ey} = \frac{\pi^2 \cdot 200 \cdot 10^9}{\left(\frac{1.0 \cdot 3.16}{31.70 \cdot 10^{-3}}\right)^2}$$

$$f_{ey} = 198 \text{ MPa}$$

SANS 10162-1 13.3.2



Designer	Auto
Date	10/07/2017
Project	Masters

Page	8
Total	11

$$\frac{C_u}{C_r} + \frac{\alpha \cdot U_{1x} \cdot M_{ux}}{M_{rx}} + \frac{\beta \cdot U_{1y} \cdot M_{uy}}{M_{ry}} \leq 1.0$$

$$\frac{5.21 \cdot 10^3}{1.10 \cdot 10^6} + \frac{0.85 \cdot 1.0 \cdot 5.46 \cdot 10^3}{100.32 \cdot 10^3} + \frac{0.85 \cdot 1.0 \cdot 0}{28.15 \cdot 10^3} \leq 1.0$$

$$0.05 \leq 1.0 \text{ OK}$$

SANS 10162-1 13.8

### Lateral torsional buckling Strength

SANS 10162-1 13.8.2  
c)

$$f_{ey} = \frac{\pi^2 \cdot E}{\left(\frac{K_y L_y}{r_y}\right)^2}$$

$$f_{ey} = \frac{\pi^2 \cdot 200 \cdot 10^9}{\left(\frac{1.0 \cdot 3.16}{31.70 \cdot 10^{-3}}\right)^2}$$

$$f_{ey} = 198 \text{ MPa}$$

SANS 10162-1 13.3.2

$$\lambda_y = \sqrt{\frac{f_y}{f_{ey}}}$$

$$\lambda_y = \sqrt{\frac{355 \cdot 10^6}{198.36 \cdot 10^6}}$$

$$\lambda_y = 1.34$$

SANS 10162-1 13.3.1

$$C_{ry} = \phi \cdot A \cdot f_y \cdot (1 + \lambda_y^{2 \cdot n})^{-1/n}$$

$$C_{ry} = 0.9 \cdot 3.82 \cdot 10^{-3} \cdot 355 \cdot 10^6 \cdot (1 + 1.34^{2 \cdot 1.34})^{-1/1.34}$$

$$C_{ry} = 515 \text{ kN}$$

SANS 10162-1 13.3.1

$$\bar{r}_o^2 = (x_o)^2 + (y_o)^2 + (r_x)^2 + (r_y)^2$$

$$\bar{r}_o^2 = (0)^2 + (0)^2 + (87 \cdot 10^{-3})^2 + (31.70 \cdot 10^{-3})^2$$

$$\bar{r}_o^2 = 8574 \text{ mm}^2$$

SANS 10162-1 13.3.2

$$f_{ez} = \left( \frac{\pi^2 \cdot E \cdot C_w}{K_z^2 \cdot L_z^2} + G \cdot J \right) \cdot \frac{1}{A \cdot \bar{r}_o^2}$$

$$f_{ez} = \left( \frac{\pi^2 \cdot 200 \cdot 10^9 \cdot 37.30 \cdot 10^{-9}}{1.0^2 \cdot 3.16^2} + 77 \cdot 10^9 \cdot 105 \cdot 10^{-9} \right) \cdot \frac{1}{3.82 \cdot 10^{-3} \cdot 8.57 \cdot 10^{-3}}$$

$$f_{ez} = 472 \text{ MPa}$$

SANS 10162-1 13.3.2

$$\lambda_{ez} = \sqrt{\frac{f_y}{f_{ez}}}$$

$$\lambda_{ez} = \sqrt{\frac{355 \cdot 10^6}{471.65 \cdot 10^6}}$$

$$\lambda_{ez} = 0.87$$

SANS 10162-1 13.3.1

$$C_{rez} = \phi \cdot A \cdot f_y \cdot (1 + \lambda_{ez}^{2 \cdot n})^{-1/n}$$

$$C_{rez} = 0.9 \cdot 3.82 \cdot 10^{-3} \cdot 355 \cdot 10^6 \cdot (1 + 0.87^{2 \cdot 1.34})^{-1/1.34}$$

$$C_{rez} = 827 \text{ kN}$$

SANS 10162-1 13.3.1



Designer	Auto
Date	10/07/2017
Project	Masters

Page	9
Total	11

$$M'_{cr} = \frac{\pi}{K \cdot L} \cdot \sqrt{E \cdot I_y \cdot G \cdot J + \left( \frac{\pi \cdot E}{K \cdot L} \right)^2 \cdot I_y \cdot C_w}$$

$$M'_{cr} = \frac{\pi}{1.0 \cdot 3.16} \cdot$$

$$\sqrt{200 \cdot 10^9 \cdot 3.84 \cdot 10^{-6} \cdot 77 \cdot 10^9 \cdot 105 \cdot 10^{-9} + \left( \frac{\pi \cdot 200 \cdot 10^9}{1.0 \cdot 3.16} \right)^2 \cdot 3.84 \cdot 10^{-6} \cdot 37.30 \cdot 10^{-9}}$$

$$M'_{cr} = 108 \text{ kNm}$$

$$M_{cr} = \omega_2 \cdot M'_{cr}$$

$$M_{cr} = 2.34 \cdot 108.21 \cdot 10^3$$

$$M_{cr} = 253 \text{ kNm}$$

$$M_{cr} > 0.67 \cdot M_p$$

$$M_{cr} > 0.67 \cdot 100.32 \cdot 10^3$$

$$M_{cr} > 67.2 \text{ kNm}$$

$$M_r = 1.15 \cdot \phi \cdot M_p \cdot \left( 1 - \frac{0.28 \cdot M_p}{M_{cr}} \right)$$

$$M_r = 1.15 \cdot 0.9 \cdot 100.32 \cdot 10^3 \cdot \left( 1 - \frac{0.28 \cdot 100.32 \cdot 10^3}{253.08 \cdot 10^3} \right)$$

$$M_r = 92.3 \text{ kNm}$$

$$M_r \leq \phi \cdot M_p$$

$$M_r \leq 0.9 \cdot 100.32 \cdot 10^3$$

$$M_r \leq 90.29 \cdot 10^3$$

NOT OK

$$M_r = \phi \cdot M_p$$

$$M_r = 90.3 \text{ kNm}$$

$$M_{ry} = \phi \cdot Z_{ply} \cdot f_y$$

$$M_{ry} = 0.9 \cdot 88.10 \cdot 10^{-6} \cdot 355 \cdot 10^6$$

$$M_{ry} = 28.1 \text{ kNm}$$

$$f_{ey} = \frac{\pi^2 \cdot E}{\left( \frac{K_y \cdot L_y}{r_y} \right)^2}$$

$$f_{ey} = \frac{\pi^2 \cdot 200 \cdot 10^9}{\left( \frac{1.0 \cdot 3.16}{31.70 \cdot 10^{-3}} \right)^2}$$

$$f_{ey} = 198 \text{ MPa}$$

$$\frac{C_u}{C_r} + \frac{\alpha \cdot U_{1x} \cdot M_{ux}}{M_{rx}} + \frac{\beta \cdot U_{1y} \cdot M_{uy}}{M_{ry}} \leq 1.0$$

$$\frac{5.21 \cdot 10^3}{514.58 \cdot 10^3} + \frac{0.85 \cdot 1.0 \cdot 5.46 \cdot 10^3}{90.29 \cdot 10^3} + \frac{0.85 \cdot 1.0 \cdot 0}{28.15 \cdot 10^3} \leq 1.0$$

$$0.06 \leq 1.0 \text{ OK}$$

SANS 10162-1 13.6

SANS 10162-1 13.6

SANS 10162-1 13.5

SANS 10162-1 13.3.2

SANS 10162-1 13.8



Designer	Auto
Date	10/07/2017
Project	Masters

Page	10
Total	11

$$M'_{cr} = \frac{\pi}{K \cdot L} \cdot \sqrt{E \cdot I_y \cdot G \cdot J + \left( \frac{\pi \cdot E}{K \cdot L} \right)^2 \cdot I_y \cdot C_w}$$

$$M'_{cr} = \frac{\pi}{1.0 \cdot 3.16} \cdot$$

$$\sqrt{200 \cdot 10^9 \cdot 3.84 \cdot 10^{-6} \cdot 77 \cdot 10^9 \cdot 105 \cdot 10^{-9} + \left( \frac{\pi \cdot 200 \cdot 10^9}{1.0 \cdot 3.16} \right)^2 \cdot 3.84 \cdot 10^{-6} \cdot 37.30 \cdot 10^{-9}}$$

$$M'_{cr} = 108 \text{ kNm}$$

$$M_{cr} = \omega_2 \cdot M'_{cr}$$

$$M_{cr} = 2.34 \cdot 108.21 \cdot 10^3$$

$$M_{cr} = 253 \text{ kNm}$$

$$M_{cr} > 0.67 \cdot M_p$$

$$M_{cr} > 0.67 \cdot 100.32 \cdot 10^3$$

$$M_{cr} > 67.2 \text{ kNm}$$

$$M_r = 1.15 \cdot \phi \cdot M_p \cdot \left( 1 - \frac{0.28 \cdot M_p}{M_{cr}} \right)$$

$$M_r = 1.15 \cdot 0.9 \cdot 100.32 \cdot 10^3 \cdot \left( 1 - \frac{0.28 \cdot 100.32 \cdot 10^3}{253.08 \cdot 10^3} \right)$$

$$M_r = 92.3 \text{ kNm}$$

$$M_r \leq \phi \cdot M_p$$

$$M_r \leq 0.9 \cdot 100.32 \cdot 10^3$$

$$M_r \leq 90.29 \cdot 10^3$$

NOT OK

$$M_r = \phi \cdot M_p$$

$$M_r = 90.3 \text{ kNm}$$

$$M_{ry} = \phi \cdot Z_{ply} \cdot f_y$$

$$M_{ry} = 0.9 \cdot 88.10 \cdot 10^{-6} \cdot 355 \cdot 10^6$$

$$M_{ry} = 28.1 \text{ kNm}$$

$$\frac{M_{ux}}{M_{rx}} + \frac{M_{uy}}{M_{ry}} \leq 1.0$$

$$\frac{5.46 \cdot 10^3}{90.29 \cdot 10^3} + \frac{0}{28.15 \cdot 10^3} \leq 1.0$$

$$0.06 \leq 1.0 \text{ OK}$$

**Member e2 has sufficient capacity**

SANS 10162-1 13.6

SANS 10162-1 13.6

SANS 10162-1 13.5

SANS 10162-1 13.8



Designer	Auto
Date	10/07/2017
Project	Masters

Page	11
Total	11

**Model sufficient for LoadCase ULS**

**Model sufficient for ALL LoadCases.**



## F. Multi-objective quantification study selected results

In this appendix the structures which corresponded with the best trade-off solution between the weight and displacement objectives are presented for each of the four examples of section 7.3.

### F.1 Plane 4-storey frame

Table F.1: Resulting sections of the plane 4-storey frame

Grouping	Assigned cross-section
Outer columns	406x140x39
Inner column	406x140x46
Floor 1 & 2	406x178x60
Floor 3 & 4	356x171x45

### F.2 Plane portal frame

Table F.2: Resulting sections of the plane portal frame

Grouping	Assigned cross-section
Columns	457x191x67
Rafters	457x191x67

### F.3 5-Bay portal frame

Table F.3: Resulting sections of the three-dimensional portal frame

Grouping	Assigned cross-section
Frame 1 & 6 columns	305x165x54
Frame 1 & 6 rafters	406x140x46
Frame 2 & 5 columns	254x146x31
Frame 2 & 5 rafters	406x140x39
Frame 3 & 4 columns	254x146x31
Frame 3 & 4 rafters	305x102x33
Frame Frame connectors	90x90x6
Cross bracing	60x60x6

### F.4 4-Storey building

Table F.4: Resulting sections of the three-dimensional 4-storey frame

Grouping	Assigned cross-section
Perimeter / outer columns	533x210x82
Internal columns	533x210x109
Perimeter/ outer beams	356x171x51
Lateral internal beams	356x171x67
Transverse internal beams	254x146x37
Bracing members	200x200x16